Estudi i implementació d'un robot controlat via Internet

Nom Estudiant: Jordi Arrieta Aymemí

Enginyeria Técnica de Telecomunicacions especialitat Telemàtica

Nom Consultor: Jordi Bécares Ferrés

Data Lliurament: 15 de Gener del 2013

Dedicat a la meva estimada xicota Delphine que m'ha donat suport en tots aquests anys d'estudi. I que crec que està tan contenta com jo de que finalment acabi la carrera, i li pugui dedicar més temps a ella.

Dedicat també al meu estimat nebot de 5 anys Jaume. Ell va sacrificar amb molt de gust un cotxe teledirigit de l'Spiderman per al meu projecte. Finalment el cotxe no l'he pogut utilitzar, però es mereix aquesta dedicatòria i un petó molt gran.

RESUM

Aquesta memòria documenta la realització i els resultats del meu treball de final de la carrera d'Enginyeria Tècnica en Telecomunicacions especialitat Telemàtica, a la Universitat Oberta de Catalunya. L'àrea escollida per el treball és la de “Sistemes encastats”.

La Universitat ens ha proporcionat un microcontrolador (LPCXpresso 1769), un mòdul Wi-Fi i un mòdul UART per a la comunicació amb un PC. Amb aquests elements havíem de crear una aplicació a escollir entre 2 propostes.

El projecte realitzat consisteix en un sistema mòbil teledirigit des de Internet. Els elements indicats en el paràgraf anterior s'han muntat sobre una plataforma, a la qual se li han acoblat un motor i unes rodes. El funcionament d'aquest sistema explicat de forma bàsica és el següent:

El mòdul Wi-Fi es connecta a un servidor d'Internet al qual va fent peticions. La resposta a aquestes peticions són números aleatoris. El microcontrolador processa aquests números i segons si són parells o senars el sistema es mou endavant o endarrere Apart s'inclouen altres funcionalitats addicionals que seran explicades més endavant.

Per programar el microcontrolador s'ha utilitzat el llenguatge C amb el suport d'un Sistema Operatiu en temps real, el FreeRTOS. L'entorn de desenvolupament, basat en el popular “Eclipse”, és proporcionat de forma gratuïta pel fabricant del microcontrolador

Aquest projecte ha significat per a mi una introducció a l'apassionant món dels microcontroladors i la robòtica. Encara que el producte final no és d'una gran complexitat, la seva realització m'ha permès veure les grans possibilitats que ofereix aquest món, i m'ha permès adquirir les bases per poder introduir-m'hi més.

Índex de continguts

Índex de figures.

1. Introducció

1.1 Justificació

Aquest projecte s'emmarca per una banda en l'àmbit dels sistemes encastats, ja que consta d'un microcontrolador que forma part d'un altre dispositiu i que realitza una funció especifica. Aquests sistemes es troben avui en dia per tot arreu, controlant una gran part dels electrodomèstics i maquines en general que hi ha al nostre voltant. Acostumen a ser sistemes petits, barats i amb baix consum energètic

Per altre banda, s'emmarca també dintre de les disciplines de l'automàtica i la robòtica. Aquestes disciplines estudien com es poden controlar sistemes i processos sense la intervenció humana. S'utilitzen per substituir les persones en processos que requereixen habilitats especials (força, precisió), per indrets massa perillosos pel esser humà, o senzillament perquè resulten més barats.

Per tant, veiem que es tracta de tecnologies àmpliament utilitzades, i amb un potencial de creixement molt gran. La tendència es a incorporar cada cop mes l'electrònica i els sistemes encastats en virtualment qualsevol objecte. I a més dotar-los de comunicacions, és el que s'anomena "Internet de les coses", a on els microcontroladors tindran un paper vital.

En concret, l'aplicació que s'ha desenvolupat, ens permet moure un objecte a distancia, a traves d'Internet. A mes, aquest objecte, que està dotat de sensors, pot interactuar amb el mon real i retroalimentar-se d'ell. Les aplicacions son innumerables, en citarem nomes algunes:

- ➔ Accés a zones no accessibles als humans, per filmar o be recollir mostres: Un exemple podria ser una central nuclear a on hi ha hagut un accident, com a Fukushima.

- ➔ Poder moure qualsevol objecte o maquinaria a distancia, sense estar-hi presents, inclús de forma programada amb anterioritat.

- ➔ Tenir algun tipus de maquinaria que realitzi una acció en resposta a un esdeveniment extern.

- ➔ Crear robots per entreteniment.

De fet, dispositius com aquest, tot i que molt més sofisticats, s'utilitzen avui en dia en molts entorns: com a armes sofisticades, per feines industrials, per cirurgia a distancia, o per simple entreteniment.

Per tant, com a resum i justificació del TFC, crec que les tecnologies implicades en el projecte tenen un gran present i futur, i es important seguir treballant i aprenent en aquest àmbit.

1.2. Descripció del projecte

El projecte consisteix en el disseny i implementació bàsic d'un robot. Es treballara a partir del següent material que es proporciona al alumne:

- ➔ microcontrolador dotat d'un processador ARM Cortex M3.

- ➔ Modul Wi-Fi RN-XV de Roving Networks.

- ➔ Modul UART CP2102 per connexió amb PC.

Al qual l'hi haurem d'acoblar un motor i unes rodes, que permetin el moviment del sistema.

Els requisits que ha de complir el producte final son els següents:

- ➔ El Robot haurà de ser capaç de fer 2 moviments bàsics: Avançar i retrocedir. No es contempla com a requisit que el moviment hagi de ser perfectament rectilini, petites desviacions en la trajectòria son tolerables.

- ➔ El Robot haurà de ser capaç també de rebre dades d'Internet i realitzar els moviments d'acord a aquestes dades. Per la connexió a Internet s'ha d'utilitzar el modul Wi-Fi proporcionat.

- ➔ Finalment, haurà de disposar d'un sistema automàtic, que li permeti de moure's fins al punt a on la cobertura de l'Accés Point sigui més gran.

1.3. Objectius

Els objectius que s'han marcat són els següents:

1.- Control del motor amb la LPCXpresso:

El primer objectiu que s'ha aconseguit amb el TFC és aprendre a controlar un motor a través del microcontrolador. Primer s'ha fet la connexió de tot el sistema, tenint en compte tant les característiques elèctriques (voltatge, corrent) com físiques (pins, cables). Després s'ha estudiat el funcionament del driver hardware que s'utilitza per comunicació entre el microcontrolador i el motor. Finalment, s'ha creat una petita aplicació que ens permet moure el motor endavant i endarrere a voluntat.

2.- Control del motor segons dada rebuda de l'Arpalab:

El segon objectiu, un cop aprés com fer funcionar el motor, és crear la funcionalitat bàsica de la nostra aplicació. Aquesta demana una dada a l'Arpalab a traves del modul Wifly. Un cop hem processat la dada, i segons si és parella o imparella, mourem el motor endavant o endarrere S'utilitzen les funcionalitats bàsiques del FreeRTOS: tasques, cues, semàfors,timers, i d'altres.

3.- Cerca de la millor cobertura:

Aquesta és una funcionalitat també requerida de l'aplicació. Quan el programa s'inicia, automàticament el robot es desplaça a la zona de major cobertura.

4.- Funcionalitats addicionals:

Amb els objectius anteriors, ja hem aconseguit implementar totes les funcionalitats requerides pel projecte. L'ultim objectiu de l'aplicació creada és poder implementar un seguit de funcionalitats addicionals, no contemplades com a requisits. Les funcionalitats que s'han pogut implementar son les següents:

- ➔ Sensor de contacte: S'ha incorporat a la part posterior del robot un sensor de contacte. Aquest sensor ens permetrà detectar quan el robot col·lisiona amb algun obstacle al retrocedir. Si es aixi, el robot avançarà de nou endavant per evitar l'obstacle.

- ➔ Botó de reset: També s'incorpora un botó de reinici. Aquest botó al ser pitjat provoca un reset instantani del sistema, i tornem a començar de nou tot el proces.

1.4. Enfocament i mètode seguit.

El TFC incloïa 2 propostes de projecte a escollir:

- Proposta 1: Compressió i estudi:

El projecte consta, en que la mota (embedded 1769) descarregui un seguit de dades de Internet, les comprimeix-hi a la mota i després envii el fitxer comprimit i les estadístiques de la compressió (nivell de compressió, temps de compressió, memòria utilitzada, ...) a Internet. El algoritme a aplicar es de lliure elecció per l'estudiant, com més eficient i òptim sigui, millor.

- Proposta 2 – Control de motors:

El projecte consisteix en afegir un petit motor i rodes a la mota, la qual pugui anar endavant i enredera (direcció no es contempla com a requisit bàsic). El sistema ha de ser capaç de rebre dades via Internet (a través del wifi) per anar endavant i enredera. També ha de disposar d'un sistema automàtic, el qual buscarà el punt on te millor cobertura wifi (cerca de l'acces point).

La primera fase, que podríem anomenar fase comuna, compren les 3 primeres PACs. Aquesta fase ens ha permès complir el següents objectius:

- ➔ Instal·lar i configurar l'entorn de programació.
- ➔ Començar a familiaritzar-nos amb el hardware que s'ens ha proporcionat, i que utilitzarem pel projecte: característiques, funcionament, comunicació entre les diferents plaques i altres.

- ➔ S’ha fet un seguit d’entregues de codi que ha pogut ser reutilitzat en gran part pel projecte final: driver Wifly i diverses llibreries.

La segona fase ha començat amb la tria de la proposta a seguir. He triat la proposta numero 2, degut a que tinc interés en l'electrònica, i m’ha semblat més interessant.

A partir d'aquí s’ha fet una planificació de la resta d’activitats fins a l’entrega final del codi, de la memòria i de la presentació. Aquesta planificació s’ha plasmat en un Diagrama de Gantt amb totes les tasques necessàries i el termini de realització de cada una. Aquest diagrama és una ajuda inestimable per poder anar acotant les diverses fites i avançar d’una manera ordenada, a més de poder comprovar en cada moment si ens estem endarrerint en la realització del projecte. Tot i això, la veritat és que al final no totes les tasques s’han pogut realitzar en l’ordre i termini establerts. En l’apartat següent es pot veure quines han estat les desviacions i les seves causes.

Respecte a la metodologia seguida, referent a la part de programació l’entorn venia ja establert. S’ha utilitzat el llenguatge C, en un entorn de programació Eclipse i amb el Sistema Operatiu en temps real FreeRTOS.

En quant a la part de maquinari, se’ns ha proporcionat el microcontrolador, el modul Wi-Fi i el modul UART a l’inici del projecte. Respecta a la resta de material, s’ha utilitzat com a base del robot una “protoboard”, a la qual se li ha afegit unes rodes i un motor. La decisió més important has estat com controlar el moviment del motor. Hi havia 2 opcions:

- ➔ Utilitzar la sortida digital d’un pin GPIO per moure el motor. El problema es que la senyal d’aquestes sortides no te prou potència per moure el motor, com he pogut comprovar.

- ➔ Utilitzar un driver de motor: Es una petita placa que entre d'altres funcions amplifica la senyal. També facilita l'us del motor, ja que amb una senzilla comanda pel port serie podem controlar direcció i velocitat de 2 motors. Aquesta ultima opció és la que s'ha utilitzat finalment.

Per la compra de material (driver motor, cables, sensors) s'ha consultat diversos catàlegs de botigues on-line d'electrònica Finalment s'han comprat a http://www.cooking-hacks.com, un proveïdor de Sparkfun a Espanya.

Per a la realització practica del projecte s'ha decidit establir petites fites per anar avançant de manera segura, testejant cada part i assegurant-nos que funciona abans de passar a la següent. En quant a la memòria, aquest esborrany s'ha anat realitzant de manera paral·lela a la programació.

1.5. Planificació

La planificació inicial es va fer dividint el projecte en un seguit de tasques. Així es pretenia poder fer implementacions parcials que facilitessin fer un seguiment acurat del desenvolupament del projecte. Aquestes implementacions parcials també haurien de permetre fer proves parcials del producte per poder detectar el més aviat possible qualsevol error. El llistat de tasques segons aquesta planificació inicial és el següent:

1.- Estudi manual LPCXpresso: com podem fer per moure un motor, característiques que ha de tenir el motor (voltatge, amperatge). (03/12/2012-05/12/2012)

2.- Selecció i compra material (motor, rodes). (06/12/2012-08/12/2012).

3.- Estudi funcionament motor i connexió a la placa (09/12/2012-10/12/2012)

4.- Prova de moviment del motor des de el microcontrolador (11/12/2012-12/12/2012)

5.- Creació funcions que permetin moure el motor en ambdues direccions. (13/12/2012-15/12/2012)

6.- Creació aplicació que mogui el motor segons comandes rebudes de l'Arpalab.(16/12/2012-19/12/2012)

7.- Testeig aplicació. (20/12/2012)

8.- Afegir funcionalitat per cerca automàtica de la millor cobertura.(21/12/2012-23/12/2012)

9.- Testeig aplicació.(24/12/2012)

10.- Estudi possibles funcionalitats addicionals. (27/12/2012-28/12/2012)

11- Implementació funcions addicionals (29/12/2012-31/12/2012)

12.- Muntatge final robot.(01/01/2013)

13.- Entrega codi final. (02/01/2013)

14.- Elaboració esborrany memòria (en paral·lel a totes les activitats anteriors).

15.- Elaboració document memòria final.(03/01/2013-14/01-2013)

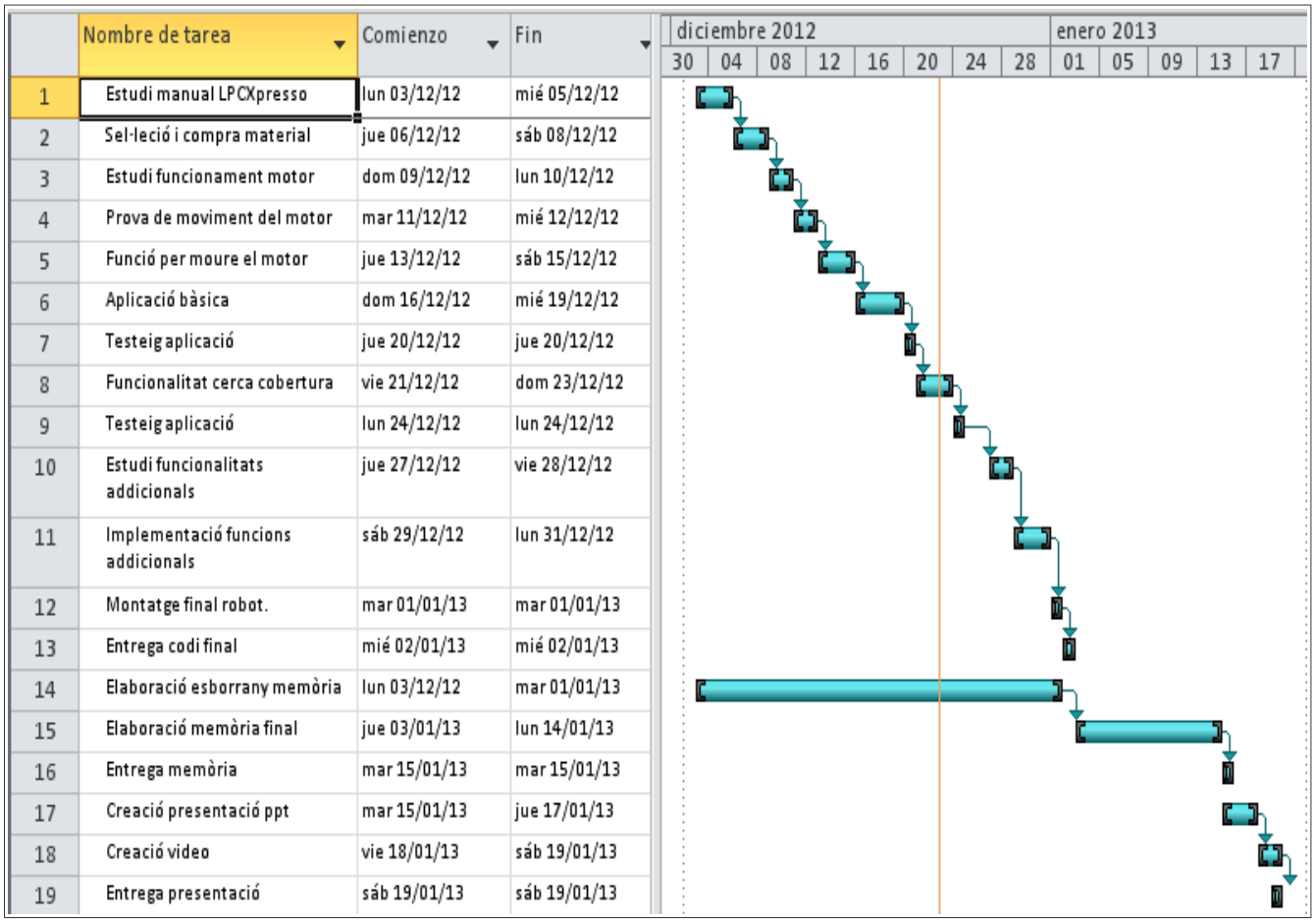
16.- Entrega memòria (15/01/2013).

17.- Creació presentació ppt. (15/01/2013-17/01/2013)

18.- Creació vídeo (18/01/2013-19/01/2013).

19.- Entrega presentació (19/01/2103)

I aquí sota ho podem veure reflexat en un Diagrama de Gantt:



	Nombre de tarea	Comienzo	Fin
1	Estudi manual LPCXpresso	lun 03/12/12	mié 05/12/12
2	Sel·lecció i compra material	jue 06/12/12	sáb 08/12/12
3	Estudi funcionament motor	dom 09/12/12	lun 10/12/12
4	Prova de moviment del motor	mar 11/12/12	mié 12/12/12
5	Funció per moure el motor	jue 13/12/12	sáb 15/12/12
6	Aplicació bàsica	dom 16/12/12	mié 19/12/12
7	Testeig aplicació	jue 20/12/12	jue 20/12/12
8	Funcionalitat cerca cobertura	vie 21/12/12	dom 23/12/12
9	Testeig aplicació	lun 24/12/12	lun 24/12/12
10	Estudi funcionalitats addicionals	jue 27/12/12	vie 28/12/12
11	Implementació funcions addicionals	sáb 29/12/12	lun 31/12/12
12	Montatge final robot.	mar 01/01/13	mar 01/01/13
13	Entrega codi final	mié 02/01/13	mié 02/01/13
14	Elaboració esborrany memòria	lun 03/12/12	mar 01/01/13
15	Elaboració memòria final	jue 03/01/13	lun 14/01/13
16	Entrega memòria	mar 15/01/13	mar 15/01/13
17	Creació presentació ppt	mar 15/01/13	jue 17/01/13
18	Creació video	vie 18/01/13	sáb 19/01/13
19	Entrega presentació	sáb 19/01/13	sáb 19/01/13

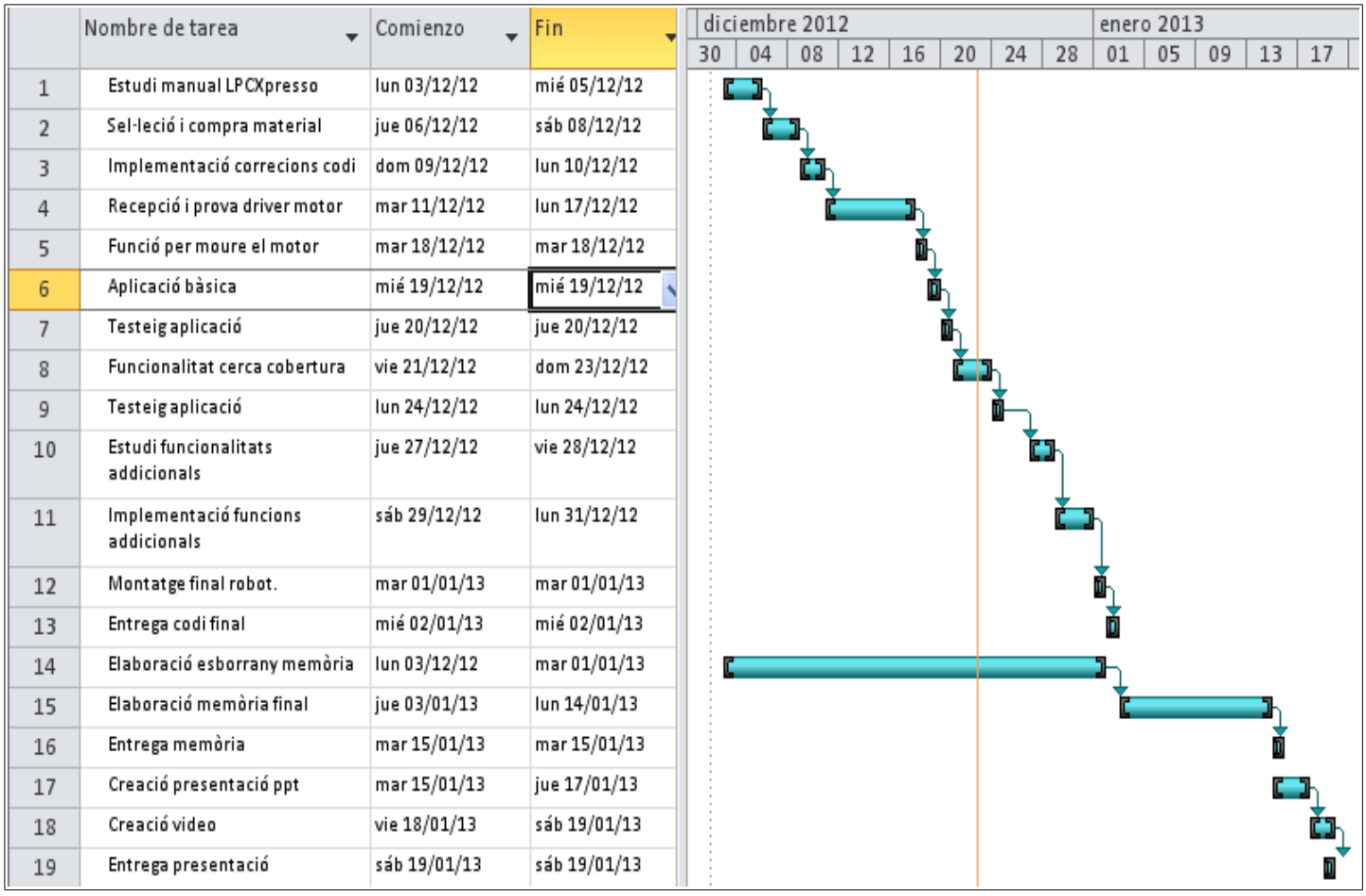
Diagrama de Gantt planificació inicial

Tot i que des de el principi s'ha intentat seguir aquest diagrama de forma estricta, la veritat és que no ha estat possible. S´han produït algunes desviacions que han provocat un petit reordenament d'algunes de les tasques. Malgrat això, ha estat possible respectar les dates d'entrega.

Els problemes que han sorgit han estat bàsicament relacionats amb la part mecànica del projecte, concretament el motor. Com que s'ha comprovat que el microcontrolador per si sol no té potència suficient per moure el motor, s'ha fet necessari comprar una placa controladora. Això ha fet que el procés de compra i recepció de material s'allargués uns dies més del previst. També la tasca d'aconseguir moure el motor des de el microcontrolador s'ha allargat, degut a problemes per fer funcionar la controladora de motor correctament. Sembla ser que ha sigut degut a problemes d'alimentació.

Degut a aquest allargament del procés de compra, s´ha produït un petit buit al inici que s´ha omplert amb una nova tasca, “Implementació correccions codi”. Aquesta tasca ha consistit en modificar el codi per incorporar suggerències del consultor, o correccions que estaven pendents. Per exemple, crear un semàfor d'exclusió per l'accés al driver del mòdul Wifly.

Finalment, a la pàgina següent podem veure un diagrama amb la temporalització real de les tasques:



	Nombre de tarea	Comienzo	Fin
1	Estudi manual LPCXpresso	lun 03/12/12	mié 05/12/12
2	Sel·lecció i compra material	jue 06/12/12	sáb 08/12/12
3	Implementació correcions codi	dom 09/12/12	lun 10/12/12
4	Recepció i prova driver motor	mar 11/12/12	lun 17/12/12
5	Funció per moure el motor	mar 18/12/12	mar 18/12/12
6	Aplicació bàsica	mié 19/12/12	mié 19/12/12
7	Testeig aplicació	jue 20/12/12	jue 20/12/12
8	Funcionalitat cerca cobertura	vie 21/12/12	dom 23/12/12
9	Testeig aplicació	lun 24/12/12	lun 24/12/12
10	Estudi funcionalitats addicionals	jue 27/12/12	vie 28/12/12
11	Implementació funcions addicionals	sáb 29/12/12	lun 31/12/12
12	Montatge final robot.	mar 01/01/13	mar 01/01/13
13	Entrega codi final	mié 02/01/13	mié 02/01/13
14	Elaboració esborrany memòria	lun 03/12/12	mar 01/01/13
15	Elaboració memòria final	jue 03/01/13	lun 14/01/13
16	Entrega memòria	mar 15/01/13	mar 15/01/13
17	Creació presentació ppt	mar 15/01/13	jue 17/01/13
18	Creació video	vie 18/01/13	sáb 19/01/13
19	Entrega presentació	sáb 19/01/13	sáb 19/01/13

Diagrama de Gantt planificació final

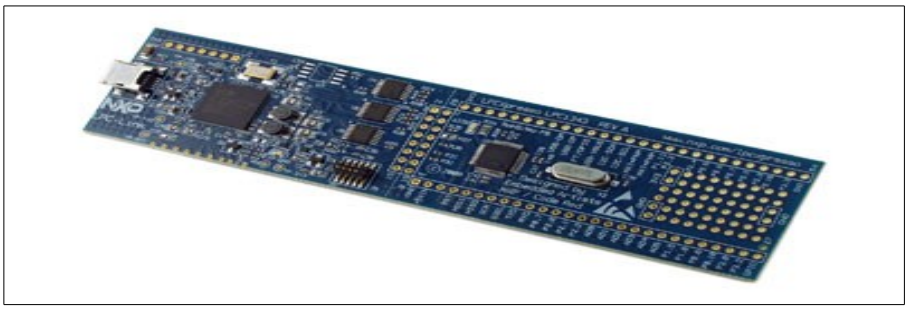
1.6. Recursos emprats

Per la realització del projecte s'han emprat una gran varietat de recursos, tant de maquinari com de programari. Alguns d'aquests recursos ens han estat proporcionats, i d'altres s'han anat comprant durant el projecte.

Recursos de maquinari

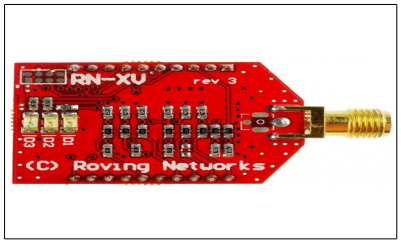
➔ Placa electrònica LPCXpresso1769, dotada amb un microcontrolador ARM Cortex M3 amb un debugger JTAG acoblat Les principals característiques són:

- 64 KB SRAM.
- 512 KB Flash.
- CPU a 120 Mhz.
- Disponibilitat diferents protocols comunicació: UART, I2c, CAN, USB, Ethernet, etc.



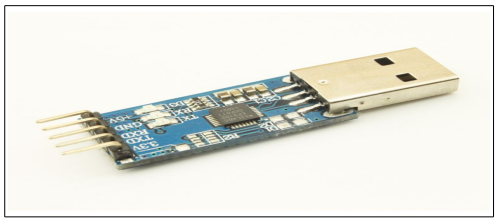
Microcontrolador LPCXpresso1769

➔ Mòdul RN-XV (Wifly) de Roving Networks. Permet connectivitat Wi-Fi utilitzant l'estandard802.11b/g. També té suport UART.



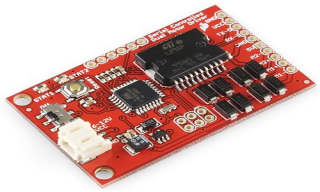
Mòdul RN-XV (Wifly)

➔ Convertidor USB-serie CP2102, per connexió de la placa i del mòdul Wifly al PC.



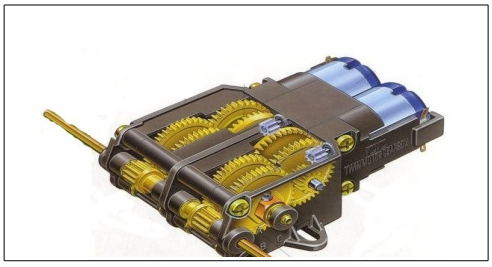
Mòdul UART CP2102

➔ Driver de motor serie ROB-09571. Amb aquesta placa podem controlar 2 motors de fins a 2 amperes cadascun. Les comandes s'envien via port serie, i permeten configurar de forma senzilla la velocitat i la direcció de gir dels motors.



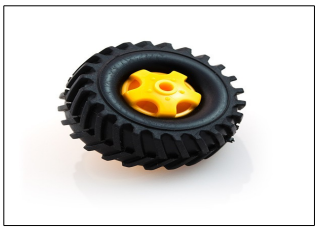
Driver motor ROB-09571

➔ Motors: Kit de l'empresa "Tamiya" format per una caixa d'engranatges amb 2 petits motors de corrent continua.



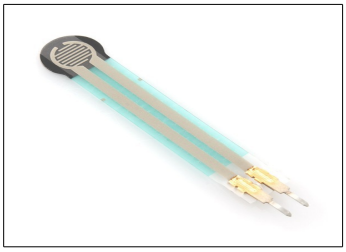
Motor i engranatges

➔ Rodes: Conjunt de 4 rodes compatibles amb el kit anterior.



Rodes

➔ Sensor de contacte: Consisteix en una resistència sensible a la pressió, que ens permet detectar quan hem fet contacte amb un obstacle.



Sensor de contacte

➔ Botó de reset: Botó per poder reiniciar el sistema.

Push button

➔ PC amb les següents característiques:

- Intel Pentium IV a 2.80 Ghz.
- Memòria RAM 1,7 GB.
- USB 2.0.

Recursos de programari

➔ Sistema operatiu PC: Linux 12.4

➔ Entorn de desenvolupament (IDE): LPCXpresso IDE. Es un entorn desenvolupat per Code Red Technologies, basat en Eclipse. Inclou diverses millores especifiques per LPC, com una llibreria optimitzada pel llenguatge C, que és el llenguatge que s'ha utilitzat per programar la aplicació.

Aquest entorn ens permet també debugar l'aplicació, aixi com descarregar-la a la flash del microcontrolador .

➔ FreeRTOS: Es un sistema operatiu de temps real al damunt del qual crearem la nostra aplicació. En realitat es pot considerar més com un kernel de temps real, optimitzat per petits sistemes embedded amb requeriments de temps real "hard". Inclou suport per 37 arquitectures de sistemes encastats. Proporciona funcions com tasques, cues, semàfors per sincronització i exclusió, i d'altres.

1.7. Productes obtinguts

Amb la realització d'aquest projecte s'han obtingut el següents productes:

- ➔ Un robot, format per una estructura amb rodes que es mou endavant i endarrere segons les dades que rep d'un servidor Internet. També te la funció de cerca de la millor cobertura. Finalment disposa d'un sensor que li indica quan hi ha un obstacle en el recorregut.

- ➔ El projecte "RobotSimple" que constitueix el codi principal de l'aplicació que controla el robot. Es el firmware que es carrega a la flash del microcontrolador. Esta creat amb el llenguatge C i amb suport del sistema operatiu en temps real FreeRTOS.

- ➔ Una serie de llibreries que implementen les funcions per accés als perifèrics i de suport a l'aplicació. Aquestes llibreries s'explicaran amb mes detalls als següents apartats.

1.8. Breu descripció dels altres capítols.

Un cop feta aquesta introducció del projecte, passem a explicar el que veurem en els següents capítols:

Primer de tot, en el segon capítol parlarem dels antecedents del projecte. Veurem quin és l'estat actual de l'art en aquesta àrea, i després farem un breu repàs de productes similars que podem trobar en el mercat.

En el capítol tercer ja veiem una descripció funcional de tot el sistema, tant a nivell programari com maquinari. Això s'explicarà molt més a fons en el quart capítol.

Els 2 següents capítols són per veure la viabilitat tècnica i la valoració econòmica del projecte.

L'últim capítol el dediquem a les conclusions sobre el projecte, a on també hi veurem una sèrie de propostes de millora.

Tanquen el projecte el glossari,la bibliografia i un annexe, a on s'explica com compilar i executar l'aplicació final.

2. Antecedents

2.1. Estat de l'art

Microcontrolador:

Per aquest projecte s'ha utilitzat un microcontrolador LPCXpresso 1769. Es tracta d'una plataforma per desenvolupament de sistemes de baix cost. Incorpora un entorn de desenvolupament gratuït basat en Eclipse. El seu nucli es un microprocessador ARM Cortex M3.

El processadors ARM tenen una arquitectura RISC de 32 bits. Avui en dia són dels mes usats en el mercat dels sistemes encastats i les tecnologies mòbils (telèfons intel·ligents, tauletes, videoconsoles). Son ideals per aplicacions de baixa potencia, degut a la seva relativa senzillesa.

Algunes de les alternatives més populars són:

- Arduino: Es un projecte de maquinari lliure, que significa que les especificacions i els esquemàtics són de domini públic. Utilitza microcontroladors ATmega de ATMEL, tot i que últimament han començat a sortir plaques amb ARM Cortex M3.

  Per programar s'utilitza el "Arduino programming language", basat en el llenguatge "Wiring", i el "Arduino development environment", basat en l'entorn "processing". Les plaques es poden comprar fetes, o es poden fer a mà, ja que es proporcionen tots els esquemàtics lliurament. Són bastant més simples que la LPCXpresso, ja que la majoria incorporen només entrades/sortides digitals i comunicació via sèrie.

- PIC: Són una família molt extensa de microcontroladors RISC de 8 i 16 bits fabricats per MicroChip Tecnology. Són molt populars en industria i també entre aficionats, degut al baix cost, disponibilitat, abundant documentació i gran quantitat d'eines gratuïtes per programació. Incorporen moltes interfases: I2C, USB, USART, A/D, PWM, Ethernet, etc.

Llenguatge de programació:

Per programar el microcontrolador s'ha utilitzat el llenguatge C. Es un llenguatge de propòsit general creat originàriament pel UNIX, tot i que es pot utilitzar en qualsevol sistema operatiu i plataforma. De fet, és el llenguatge més utilitzat per crear sistemes operatius i compiladors, encara que s'utilitza per tot tipus d'aplicacions.

Es considera un llenguatge de nivell mig, ja que tot i tenir característiques d'un llenguatge d'alt nivell, també permet treballar a més baix nivell. Per exemple, es pot accedir directament a posicions de memòria amb el apuntadors, o també es poden manejar bits de forma individual.

Altres llenguatges que s'acostumen a utilitzar per programar microcontroladors són:

- Assemblador: Es un llenguatge de baix nivell. Cada instrucció de assemblador correspon a una instrucció de codi màquina. Amb aquest llenguatge poden manejar directament els registres del processador, al ser de baix nivell. Per tant, és molt més eficient que qualsevol altre llenguatge. Els desavantatges són que els programes són més llargs d'escriure, i es necessita tenir un gran coneixement tant de programació com del processador que estem programant. A més, els programes s'escriuen específicament per un processador, no són portables.

- BASIC: Es una família de llenguatges d'alt nivell. El seu ús es bastant senzill, de fet originàriament va sorgit per educació i per usuaris poc experts. Hi ha variants del llenguatge original per poder programar microcontroladors. Al ser un llenguatge d'alt nivell, no té l'eficiència dels altres dos.

2.2. Estudi de mercat

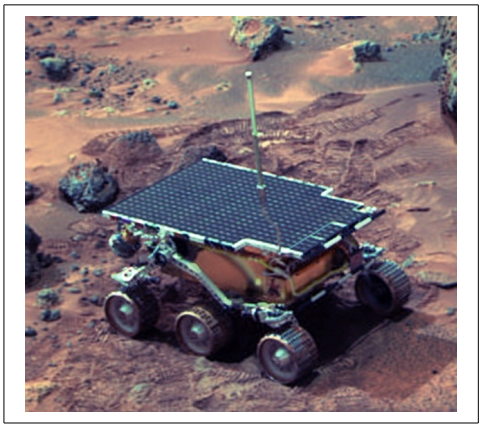
Actualment existeixen molts robots com el del present projecte. O sigui, dispositius dotats de moviment, que es poden connectar a alguna xarxa de comunicacions per poder ser controlats a distancia, i amb sensors per poder interactuar amb el seu entorn.

A mode d'exemple farem una petita introducció d'alguns d'ells:

- Mars Rover: Son vehicles d'exploració enviats al planeta Mart en diferents missions de la NASA. Aquests vehicles, un cop el coet que els transporta ha aterrat, es desplacen per la superfície del planeta de forma autònoma. Estan comunicats amb una estació base, que pot ser un satèl·lit orbitant Mart o un altre dispositiu arribat en la mateixa missió. També poden tenir comunicació directe amb la Terra. D'aquesta manera se'ls hi poden enviar comandes per controlar-los i dirigir-los cap a llocs d'interès. Disposen de rodes per desplaçar-se i sensors i càmeres per detectar obstacles.

  L'alimentació d'aquests vehicles es fa de diverses formes: Bateries de liti, bateries solars, o fins i tot aprofitant el calor generat per un isòtop radioactiu.

  S'han enviat diversos d'aquests artilugis. Un d'ells es el "Sojourner Rover", enviat a Mart amb el "Mars Pathfinder", i que podem veure a la foto a la pàgina següent. Actualment es troba en actiu el "Curiosity", des de l'Agost del 2011.



Sojourner Rover

➔ Robots de seguretat i defensa: L'empresa i-Robot, fabricant dels coneguts robots aspiradors "Roomba", tambe fabrica altres tipus de robot per seguretat i defensa. Entre ells el "iRobot-110-FirstLook", utilitzat per unitats d'infanteria d'alguns exercits. Es controla per radio i permet investigar edificis, túnels i llocs de difícil accés. Esta dotat d'infrarojos per a escenaris de baixa lluminositat. Es petit i lleuger, i fins i tot es pot llençar des de una alçada d'uns 5m.

Com que ha de marxar sobre terrenys irregulars, enlloc de rodes utilitza erugues, com els tancs.

iRobot-110-FirstLook

➔ Robots de vigilància en interiors: L'empresa RoboServ, entre d'altres productes, comercialitza robots per a vigilància domestica. Un exemple es el Wifibot. Estan dotats de càmera de vídeo i de sensors de proximitat per infraroig. Disposen de nombroses interfícies, com Ethernet, RS232, I2C, USB. També es poden controlar a distancia per Internet.

Wifibot

Com podem veure, aquests robots són molt més complexes que el que presentem en aquest projecte. Malgrat això, la base és la mateixa. Són dispositius dotats de moviments i de sensors de diferents tipus per interaccionar amb el entorn. També tenen intel·ligència per prendre decisions. El seu moviment pot ser programat o dirigit en temps real a través d'Internet o ràdio.

3. Descripció funcional

3.1. Sistema total

Sistema físic:

Primer farem una descripció bàsica del sistema físic:

Es disposa d'una plataforma sobre 4 rodes, que són mogudes per 2 motors. A sobre la plataforma hi trobem els diferents dispositius: Microcontrolador, mòdul Wifly, la controladora del motor i els sensors. Aquesta plataforma esta connectada amb cables amb un mòdul UART que es troba endollat al PC. Aquesta connexió serveix per transmissió de dades al PC a través del port sèrie.

Aqui podem veure una foto del robot:

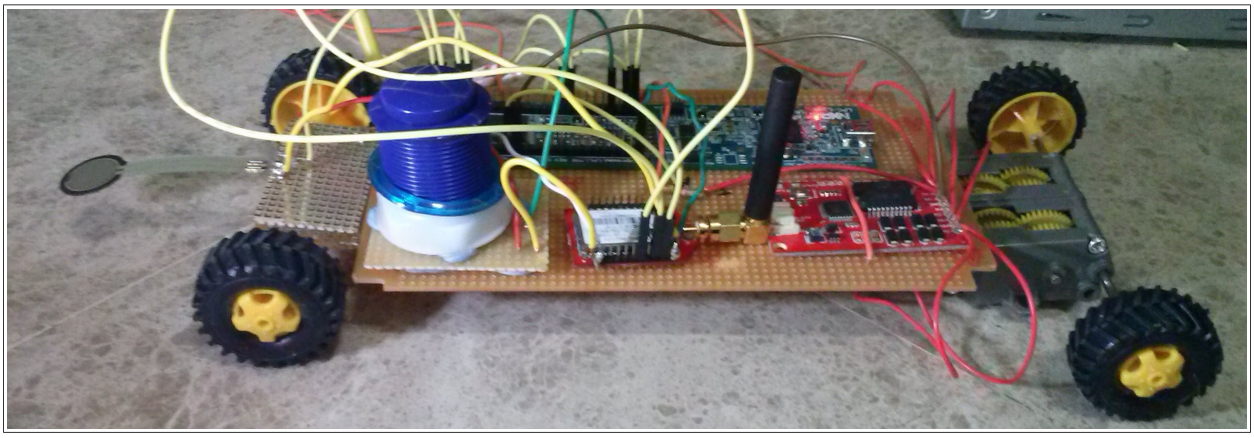


Foto del Robot lateral

El funcionament del sistema, que anomenarem robot, és el següent:

- ➔ Al arrencar, el primer que farà el robot és connectar-se a la xarxa sense fils que hem configurat.

- ➔ Un cop connectat, es desplaçarà al lloc a on la senyal del punt d'accés sigui més forta, o sigui cercarà el punt a on la cobertura sigui més gran.

- ➔ Acte seguit, el robot iniciarà un cicle de forma indefinida. Primer, farà una petició a un servidor web. Aquest servidor implementa una aplicació anomenada “arpalab”, que retorna una dada aleatòria de tipus *double* en resposta a aquesta petició.

- ➔ Al rebre la dada, el robot comprovarà si és parella o senar. Si és parella, es desplaçarà endavant. Si és senar, es desplaçarà endarrere El sistema està equipat amb uns sensors que permetran detectar obstacles, i actuar en conseqüència invertint el sentit de la marxa.

- ➔ Un cop acabat el desplaçament, es torna a reiniciar el cicle.

- ➔ Paral·lelament, el robot utilitzarà la connexió al PC a través de la UART, per mostrar per consola un log de les accions que va realitzant.

- ➔ Es disposa també d'un botó, el qual ens permet fer un reset del sistema en qualsevol moment.

Diagrama de blocs de l'aplicació:

A la pàgina següent tenim el diagrama de blocs de l'aplicació. S'han utilitzat colors per distingir de forma conceptual els diferents mòduls:

- ➔ Amb color negre els dispositius físics.
- ➔ Amb color blau els mòduls funcionals.
- ➔ Amb color vermell els arxius del projecte principal.

Acte seguit passem a fer una explicació bàsica del diagrama:

- ➔ L'aplicació principal, que consta de 2 tasques i una cua, és la que s'encarrega de realitzar totes les tasques que descriuen el funcionament del sistema com hem vist a l'apartat anterior.
- ➔ Per accedir al servidor d'Internet, utilitza el mòdul “arpalab”. Aquest mòdul, per fer la petició al servidor, es connecta al mòdul Wifly a través del driver.
- ➔ Per realitzar els desplaçaments, es donen les ordres al mòdul “Accés motor”, que actua sobre els 2 motors. Aquest mòdul rep els inputs del sensor, la qual cosa li permet detectar els obstacles.
- ➔ Finalment,el mòdul UART és el que ens permet mostrar per la consola del PC un registre de totes les accions que anem realitzant.

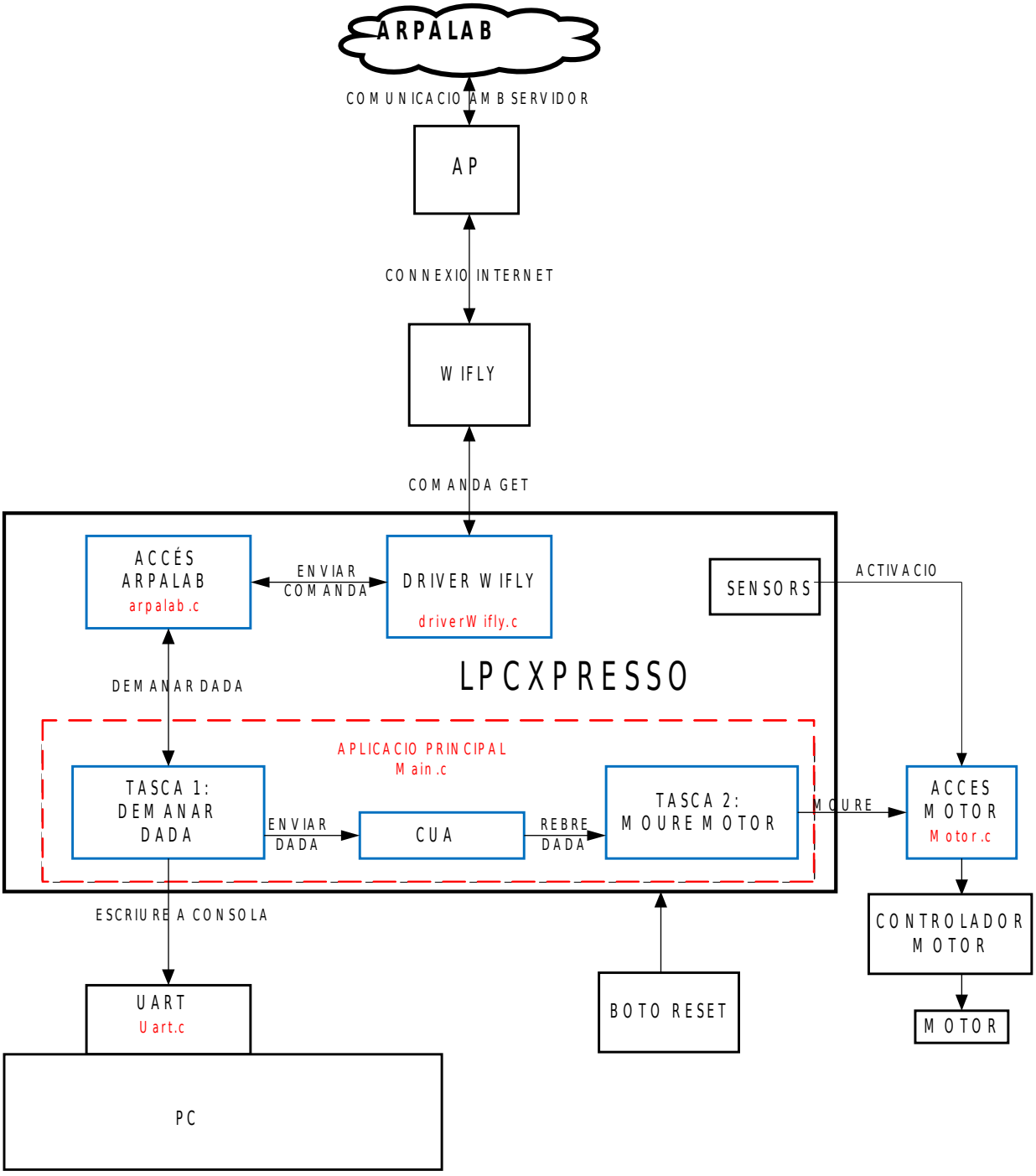


Diagrama de blocs de l'aplicació

Com és la xarxa (posició i comunicació a través de la xarxa):

El sistema necessita connectar-se a Internet per poder fer la petició de dades al servidor "Arpalab". Aquesta connexió es realitza del següent mode:

- ➔ Es disposa d'un mòdul Wi-Fi, anomenat Wifly, connectat al microcontrolador a través d'un port sèrie.
- ➔ El mòdul Wifly es connecta sense fils amb el punt d'accés, que és un Smartphone LG Optimus L5.
- ➔ El punt d'accés es connecta a Internet a través de la xarxa 3G.

Per altre banda, i com hem vist ja abans, el microcontrolador es connecta també amb un PC per mostrar un log de les accions que es realitzen. Aquesta connexió es realitza de la següent forma:

- ➔ El microcontrolador es comunica a través del port sèrie amb un mòdul UART.
- ➔ El mòdul UART està connectat a un port USB del PC.

3.2. PC (Disseny de la interfície d'usuari)

El PC,com ja s'ha dit, s'utilitza exclusivament per mostrar per pantalla un petit registre de les accions que realitza el sistema. Per fer això, només necessitem connectar-nos al port sèrie obrint una sessió amb un programa tipus "Putty" o "Cutecom".

L'usuari no pot interaccionar amb el sistema.

3.3. Disseny de l'aplicació

L'aplicació consisteix en un projecte principal, anomenat “RobotSimple”, i un conjunt de llibreries. Aquestes llibreries s'utilitzen bàsicament per accedir als diferents perifèrics, i per implementar diverses utilitats que seran cridades des de diferents punts de l'aplicació.

A continuació farem una descripció bàsica del projecte “RobotSimple”:

Aquest projecte consta de 3 arxius:

- “main.c”: Es la funció principal de l'aplicació. Consta de 2 tasques que realitzen les següent funcions:

    1. **“dataTask”:** Aquesta tasca és la que s'encarrega de demanar les dades al servidor “Arpalab”. Un cop rep una dada, l'envia a una cua perquè la pugui llegir l'altre tasca.

    2. **“MotorTask”:** Aquesta tasca el primer que fa és llegir una dada de la cua, que representa una xifra. Després, comprova si és parella o senar. Si és parella, dona l'ordre al motor perquè es mogui cap endavant. Si és senar, dona l'ordre al motor perquè es mogui endarrere

- “arpalab.c”: Crea una capa d'abstracció entre l'aplicació principal i el servidor “arpalab”. Implementa 2 funcions:

    1. **“reqData”:** Fa la petició d'una dada al servidor a través del mòdul Wifly.

2. **“lastDigit”:** Aquesta funció rep un número en format *string*, i retorna l'últim digit en format enter. L'utilitza la tasca “motorTask” per poder comprovar si la dada llegida de la cua és un número parell o senar.

➔ “motor.c”: Aquesta funció és la que s'encarrega de moure el motor per avançar o retrocedir. Consta de les següents funcions:

1. **“motorInit”:** Inicialitza la connexió sèrie amb la controladora del motor. També configura el pin a on es connecta el sensor com a pin d'entrada de senyal.

2. **MotorMove i MotorMoveActiveDelay:** S'encarreguen de moure el motor endavant i endarrere En el primer cas, també comprova el sensor per si ha detectat alguna col·lisió, i en cas afirmatiu retocedeix per fugir de l'obstacle. La diferència entre les 2 funcions es veuraà al següent apartat.

Com hem dit, l'aplicació consta també d'un conjunt de llibreries, que explicarem de forma molt breu a continuació:

➔ FreeRTOS_Lybrary: Inclou els arxius que formen el Sistema Operatiu de temps real FreeRTOS.

➔ CMSISv1p30_LPC17xx: facilita l'accés als registres de configuració dels diferents perifèrics interns al LPCXpresso 1769.

➔ UART: Implementa les funcions necessàries per la comunicació sèrie amb els diversos perifèrics externs.

- ➔ Printf: Implementa funcions per enviar pels ports sèrie diferents estructures de dades.

- ➔ DriverWifly: Es el driver per poder comunicar-se amb el mòdul Wi-Fi.

- ➔ UtilsLPC17xx: Implementa una sèrie d'utilitats que són utilitzades en diferents parts de l'aplicació.

- ➔ GPIO: Funcions per poder utilitzar les entrades i sortides digitals.

Les 3 primeres llibreries ens han sigut proporcionades. La resta són de creació pròpia

4. Descripció detallada

4.1 Sistema físic:

Com ja hem mencionat, el sistema físic consisteix en un robot que es pot moure de forma autònoma. Aquest robot s'ha creat afegint un motor i unes rodes al microcontrolador i perifèrics.

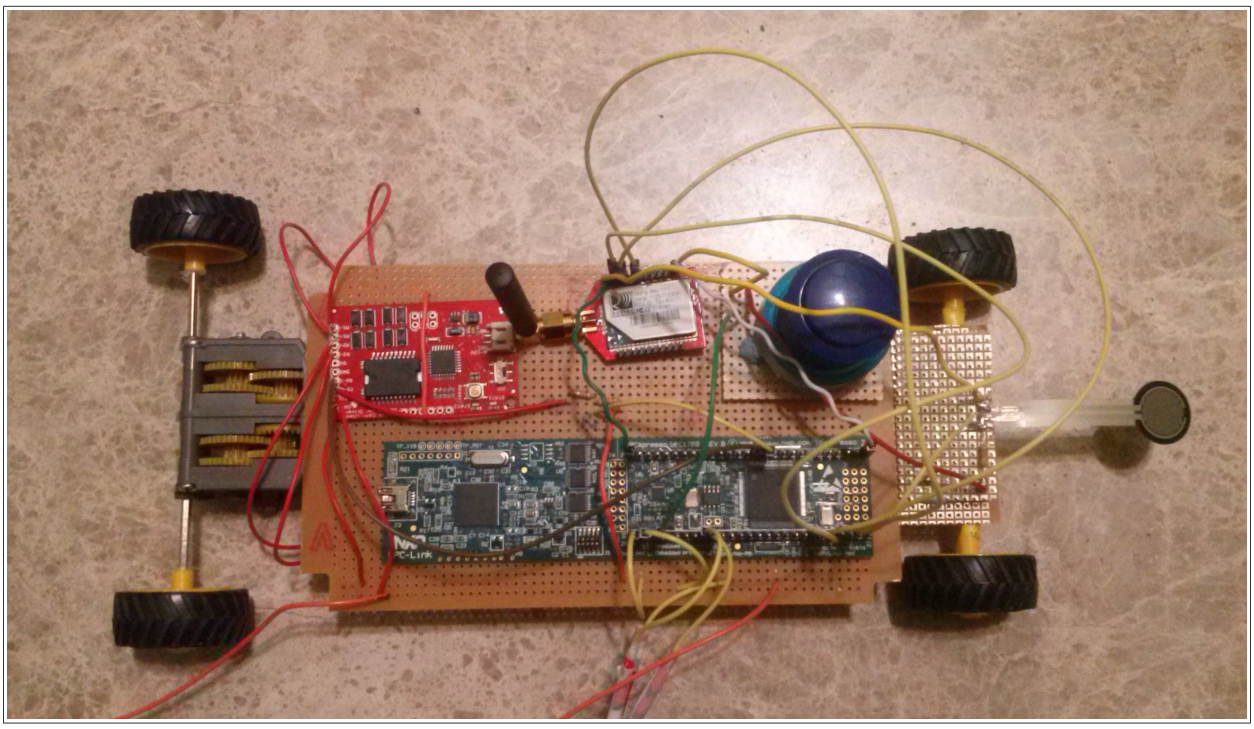


Foto aeria del Robot.

A continuació farem una descripció detallada de tots els elements:

Base:

Tots els elements que formen el sistema, excepte el mòdul UART, es troben situats sobre la mateixa plataforma. Per fer aquesta plataforma s'ha utilitzat una “protoboard”, ja que és lleugera i permet integrar en un mateix espai els diferents elements amb totes les connexions i soldadures necessàries

D'aquesta plataforma surt un cable amb 3 fils que es van al mòdul UART connectat al PC, amb la següent funció:

- ➔ 2 fils són per a la transmissió i recepció de dades sèrie entre el microcontrolador i el PC. Aquesta transmissió és necessària per mostrar per consola un logs de totes les activitats que va realitzant el Robot.

- ➔ L'altre cable és el GND de l'UART. S'ha de posar en comú amb el del LPCXpresso, per tenir els mateixos nivells de voltatge de referència.

Sistema motor:

Al principi s'havia pensat en aprofitar motor, rodes, i xassís d'un cotxe teledirigit per control remot. Però m'he trobat amb el problema de que no teníem suficient potència per moure el motor, i tampoc ens cabien tots els elements.

Finalment, s'ha decidit comprar un kit de l'empresa “Tamiya” que consisteix en una caixa d'engranatges amb 2 petits motors de corrent continua. Apart s'han comprat també 4 rodes compatibles amb el kit.

Microcontrolador :

Es el cor del sistema. Es una placa LPCXpresso 1769, amb un processador ARM Cortex M3. Es ideal per aplicacions de baix cost i baix consum. Malgrat això, inclou nombrosos perifèrics i suport per diversos busos de comunicació: UART, USB, Ethernet, I2C, CAN, ADC, entre d'altres. També inclou un programador JTAG per USB, per facilitar les tasques de programació i depuració. Per tot això, és un microcontrol·lador molt adequat per aquest projecte.

La comunicació amb el perifèrics externs és a través d'UART. Això s'ha fet així perquè la comunicació sèrie és un estàndard establert des de fa molt de temps, la majoria de dispositius l'incorporen i el seu ús es bastant senzill, requerint només de 2 cables per a la transmissió de dades.

Mòdul Wi-Fi:

Per a la connexió a Internet se'ns ha proporcionat un mòdul RN-XV (Wifly) de Roving Networks. Es un mòdul molt adequat per aquest projecte, ja que és de mida petita, i ens permet connectivitat sense fils sota l'estàndard 802.11b/g. Per comunicar-se amb el microcontrolador s'ha utilitzat el port sèrie.

Els principals pins que hi trobem són:

- ➔ Alimentació de 3.3v, que ens ve donada per la LPCXpresso.
- ➔ Tx i Rx per a la comunicació amb el microcontrolador.
- ➔ Pin de RESET, per poder reiniciar-lo des de el micro.

Modul UART:

La controladora CP2102 és un pont UART-USB que ens permet la comunicació sèrie amb el PC. A la part del PC es connecta a un port USB 2.0. Per utilitzar-ho hem d'iniciar una sessió amb un programa tipus “putty”, configurant tots els paràmetres de connexió (baudRate, paritat, etc). Un cop iniciada la sessió, ens permetrà visualitzar per consola els missatges que li envii la LPCXpresso.

El pins que utilitzem són els següents:

- ➔ Tx i Rx per comunicar-nos amb el microcontrolador.
- ➔ GND.

Controladora de motor:

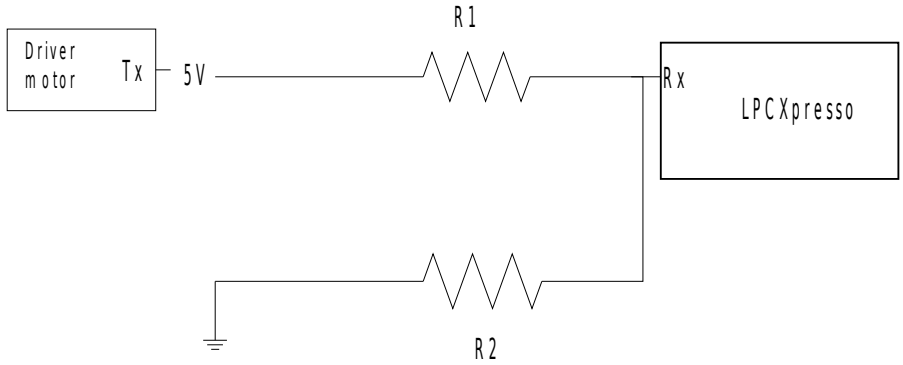
Per un control senzill dels motors, podem utilitzar alguna de les entrades/sortides digitals del microcontrolador. El problema és que la senyal de sortida no té suficient força per moure els motors. Per això s'ha optat per utilitzar una controladora de motor, la ROB-09571.

Aquesta controladora ens permet controlar 2 motors alhora, configurant la direcció de gir i la velocitat. Això últim s'aconsegueix a través de PWM. També té protecció contra excés de corrent, parant els motors si detecta aquesta situació. Incorpora els següents pins:

- ➔ Alimentació.
- ➔ Tx i Rx per a la comunicació sèrie amb el microcontrolador.

➔ M1+, M1-, M2+, M2-: Hi connectem els 2 motors.

El nivell de senyal que utilitza per a la comunicació sèrie és de 5v. Tot i que en principi la LPCXpresso pot treballar amb aquest nivell, s'ha preferit disminuir-ho per seguretat. Això s'ha fet amb el següent divisor de tensió:



Divisor tensió connexió sèrie microcontrolador – driver motor

Volem reduir la tensió al pin de Rx a 3-3,3v aproximadament. Per una intensitat de 0,30mA, adequada per a la LPC, tenim:

$R1 = V / I = 2 / 0,3 \times 10^{-3} = 6k$
$R2 = V / I = 3 / 0,3 \times 10^{-3} = 10k$

Les resistències que teníem disponibles de valor més aproximat són de 6,7KΩ i 12KΩ. En aquest cas:

$V1 = 5 \times 6,7 / (12 + 6,7) = 1,8v$

Vrx = 5 – 1,8 = 3,2v

Ens queda un valor de senyal a l'entrada de la Rx de la LPCXpresso de 3,2v.

Sensor de contacte:

Al sistema se li ha acoblat també un sensor de contacte que detecta quan el Robot xoca amb algun obstacle. Aquest sensor és una resistència sensible a la pressió. En estat normal, la seva resistència és superior als 10MΩ. Quan se li aplica pressió, la seva resistència pot baixar fins als 2,5KΩ. Així, si la col·loquem a la part davantera, al xocar amb un obstacle es doblegarà, el microcontrolador detectarà la variació de la resistència i actuarà en conseqüència.

Aquest sensor s'alimenta amb 3,3v, i es connecta a una entrada / sortida digital de propòsit general. Com que en estat normal la seva resistència és molt elevada, es comporta com un circuit obert. Això fa que l'entrada sigui indeterminada, i puguin produir-se errors de lectura. Per evitar això hem col·locat en paral·lel una connexió a terra a través d'una resistència “pull-down” de 33kΩ. Així en estat normal l'entrada és un “0” lògic.

Botó de reinici: El Robot consta també d'un botó que al ser pitjat reinicia instantàniament el sistema. Es connecta al pin de RESET, actiu per nivell baix.

4. 2 Disseny aplicació:

A continuació passarem a descriure amb més detall el disseny de l'aplicació que es carregarà al microcontrolador.

Com ja hem dit, l'aplicació consta d'un projecte principal, anomenat “RobotSimple”. Aquest projecte està format per 3 arxius, que passem a descriure tot seguit:

1. “main.c”:

Com el seu nom indica, aquest és l'arxiu principal. Bàsicament consta de 2 tasques que s'executen en un bucle indefinit. Aquestes tasques necessiten compartir informació entre elles. Per fer-ho, utilitzem una de les funcions que ens proporciona el FreeRTOS: les cues. Vegem com actuen aquestes tasques:

dataTask: Aquesta tasca realitza les següents funcions:

- ➔ Primer de tot demana una dada al servidor “Arpalab”. Això ho fa fent una crida a la funció “reqData()”, proporcionada per l'arxiu “arpalab.c”. El servidor respon enviant un número enter aleatori de 16 bits sense signe.

- ➔ Aquesta dada l'envia a la cua, utilitzant la funció “xQueueSend”, proporcionada pel FreeRTOS.

MotorTask: Realitza les següents funcions:

- ➔ Llegeix una dada de la cua. Utilitza la funció “xQueueReceive” del FreeRTOS per fer-ho.

- ➔ El número llegit de la cua es processa, de tal forma que si és parell es dona l'ordre d'avançar, i si és senar es dona l'ordre de retrocedir. Per donar l'ordre s'utilitza la funció “motorMove”, proporcionada pel “motor.c”.

Cal tenir en compte que el número que es llegeix del servidor i es comparteix a través de la cua, s'emmagatzema en format de cadena de caràcters. La funció “lastDigit”, del “arpalab.c”, ens retorna l'últim digit del número en format int, així és possible comprovar si és parell o senar, mirant si és divisible entre 2.

La segona tasca, “motorTask”, té una prioritat més alta que la primera. Així ens assegurem que, tan bon punt rebem una dada del servidor, la processarem per poder moure el robot.

L'arxiu **main.c** inclou una funció per cercar el punt a on tenim la millor cobertura, **searchReception()**. A continuació veiem el diagrama de flux de l'algorisme que utilitza la funció per acomplir la cerca:

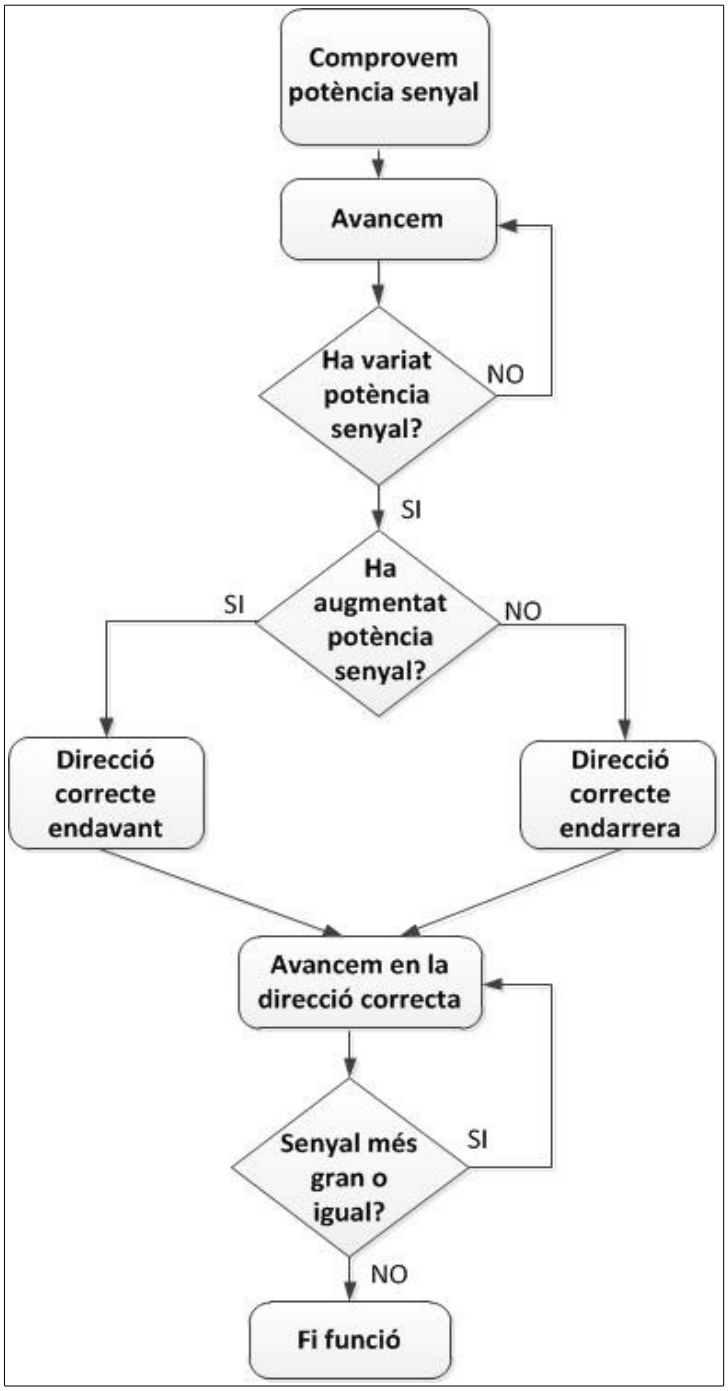


Diagrama de flux funció cerca cobertura

El que fem es avançar fins que la senyal canvia. Si la senyal ha augmentat, la direcció correcta és endavant. Si ha disminuït, es endarrere. Llavors seguim movent-nos en la direcció correcte mentre la senyal no varii o vagi augmentant.

Finalment tenim la funció main(), que realitza les següents funcions:

- ➔ Fa totes les inicialitzacions necessàries:comunicació UART amb Wifly i consola, funció Printf i motor. Després veurem amb més detall en que consisteixen aquestes inicialitzacions.

- ➔ Crida a la funció connectInit(), del mòdul “arpalab.c”. Aquesta funció inicia la connexió a Internet. També la veurem més a fons després.

- ➔ Crea les tasques i inicialitza el “scheduler”.

2. “Arpalab.c”:

Com ja hem dit, aquest mòdul crea una capa d'abstracció entre l'aplicació principal i el servidor “arpalab”. Implementa les següents funcions:

- ➔ connectInit(): Aquesta funció és la que s'encarrega de iniciar el Wifly i la connexió Internet. Per fer-ho crida a les següents funcions del driver de la Wifly:

  - wiflyConnect(): Reseteja la Wifly i entra en mode comanda.
  - WiflyNetAccess(): Fa que la Wifly es connecti a la xarxa que tenim configurada.

- ➔ reqData(): Funció que utilitzem per sol·licitar les dades al servidor. Utilitza també el

driver de la Wifly. Crida a la funció “wiflySendHTTP”, que envia una petició GET al servidor.

➔ lastDigit: Aquesta funció pren un número emmagatzemat com una cadena de caràcters i retorna l'últim dígit en format *int*. La utilitzem per poder calcular si el número és parell o senar.

3. “motor.c”:

Aquest mòdul és el que s'encarrega del control del motor. Ens proporciona les següents funcions:

➔ motorInit(): Inicialitza la comunicació UART amb la controladora del motor. També configura el pin a on es connecta el sensor de contacte, amb una crida a la funció GPIOSet de la llibreria GPIO.

➔ MotorMove(): S'encarrega de donar l'ordre a la controladora perquè mogui el motor per avançar o retrocedir.

La comunicació amb la controladora és mitjançant comandes per port sèrie. Una comanda consta de 4 caràcters:

- Un número que identifica el motor (1 o 2).
- Un caràcter per indicar la direcció (“f” endavant, “r” endarrere).
- Un número per indicar la velocitat del 0 al 9. 9 és velocitat màxima, mentre que 0 significa motor parat.

- Caràcter de retorn, “\r”.

Com a exemple, per indicar que volem moure el motor 2 endavant a màxima velocitat enviaríem la cadena “2f9\r”.

Si hem avançat, la funció comprovarà també si hem topat amb algun obstacle. Si és el cas, al pin a on tenim connectat el sensor tindrem un nivell alt de senyal. Per llegir-ho utilitzarem la funció GPIORead de la llibreria GPIO.

➔ MotorMoveActiveDelay(): Realitza exactament la mateixa funció que l'anterior. Però utilitza esperes actives enlloc de passives. Quan estem executant la funció de cerca de la millor cobertura, encara no hem creat les tasques. Les esperes passives s'utilitzen dintre de les tasques, i consisteixen en suspendre una tasca durant el temps que dura l'espera. Per tant, si encara no s'han creat les tasques, no es poden utilitzar esperes passives. Per això s'ha creat aquesta funció.

Apart de l'aplicació principal, l'entrega final inclou una sèrie de llibreries de funcions:

Uart:

Aquesta llibreria ens ha sigut proporcionada. Inclou les funcions per la comunicació del LPCXpresso amb els dispositius perifèrics a través dels 3 mòduls UART. Ens permet que el microcontrolador pugui enviar i rebre dades a la Wifly, a la controladora de motor i al PC seguint el protocol sèrie.

A continuació explicarem amb una mica més de detall les llibreries de creació pròpia:

Printf:

Aquesta llibreria utilitza les funcions de la llibreria Uart per poder enviar dades pels port sèrie amb un format determinat. Implementa les següents funcions:

➔ printfInit(): Crea un mutex o semàfor d'exclusió mútua, que utilitzaran la resta de funcions per evitar l'accés concurrent de diferents tasques.

➔ PrintText(): L'utilitzem per enviar un text pel port que l'indiquem

➔ PrintTextNum(): Envia un text més un número pel port que l'indiquem.

GPIO:

Aquesta llibreria ens permet utilitzar les entrades/sortides digitals o GPIO (General Purpose Input/output). En concret les de la part superior del port 0 (P.16 a P.30). Recordem per exemple que el sensor de contacte utilitza una d'aquestes entrades per indicar al microcontrolador quan hem topat amb un obstacle. Les funcions que inclou són:

- ➔ GPIOSet(): Ens permet indicar quin pin volem utilitzar, i si serà una entrada o una sortida.

- ➔ GPIOWrite(): Ens permet escriure un valor en una sortida. Aquest valor serà un “0” o un “1”, ja que estem parlant de sortides digitals.

- ➔ GPIORead(): Ens permet llegir el valor que té una entrada digital.

Per tal de poder implementar aquestes funcions, necessitem accedir a un seguit de registres del microcontrolador, que són específics per GPIO. Aquests registres són:

- PINSEL: En aquest registre escollim quin pin utilitzarem, i quina de les funcions que té el pin associades necessitem. En efecte, molts dels pins de la placa LPCXpresso, poden realitzar més d'una funció. Per tant abans d'utilitzar-los hem d'especificar quina funció volem que realitzi.

  Els PINSEL són registres de 32 bits, 2 bits per cada pin, que ens permet escollir 1 de les fins a 4 funcions possibles que pot tenir el pin.

- FIODIR: Aquesta registre serveix per configurar el pin com entrada (posant un “0” al bit corresponent) o sortida (posant-hi un “1”).

- FIOSET: Per escriure un “1” a la sortida. Hem de posar el bit corresponent a “1”.

- FIOCLR: Per escriure un “0” a la sortida. Hem de posar el bit corresponent a “1”.

- FIOPIN: Serveix per llegir el valor actual d'un pin d'entrada. Abans hem de posar a “0” el bit corresponent al pin en el registre FIOMASK.

DriverWifly:

Com el seu nom indica, aquesta llibreria implementa un driver per poder accedir a les funcionalitats més importants del mòdul Wifly Tot seguit passarem a comentar les funcions més importants per aquest projecte:

- ➔ wiflyConnect(): Inicialitza el mòdul per poder començar a treballar amb ell. Els passos que segueix són:
  - Primer reseteja la Wifly, així podem començar des de un estat conegut. Configurem un pin GPIO del microcontrolador per poder enviar el patró de senyal necessari (“0” - pausa mínima de 160us - “1”).
  - Després enviem l'ordre “$$$” per poder entrar en mode comanda.
  - Finalment creem el semàfor d'exclusió per evitar-hi l'accés concurrent.
- ➔ WiflyNetAccess(): Ens permet connectar-nos a la xarxa que vulguem. Prèviament haurem d'indicar el nom de la xarxa, el mode d'autenticació i la contrasenya.
- ➔ wiflyGetRSSI(): Ens retorna la potència de la senyal que rebem del punt d'accés. Aquesta funció ens serà necessària per la funcionalitat de recerca de cobertura.
- ➔ wiflySendHTTP(): Aquesta funció ens permet enviar comandes HTTP al servidor “Arpalab”. L'utilitza la tasca “dataTask” cada cop que sol·licita una dada. Hem de passar-li a la funció la comanda que volem enviar

Aquestes funcions escolten també la resposta que dona el mòdul Wifly, per saber si la comanda ha tingut èxit o ha fallat.

utilsLPC17xx.c

Ens proporciona una sèrie d'utilitats que són cridades des de diferents arxius del projecte:

- delayMs() i passiveDelayMs(): Introdueixen un retard o espera en l'execució d'una funció. En el primer cas es tracta d'una espera activa, en el segon d'una espera passiva La diferència és que l'espera activa crea un bucle buit de 7140 iteracions per mili-segon, per tant ocupa temps de processador. En l'espera passiva, la tasca deixa l'estat “run” i passen a executar-se altres tasques, per tant el processador no queda ocupat en una tasca inútil

  En principi l'espera activa és sempre preferible. El que passa es que en algunes situacions no es pot utilitzar. Per exemple, si volem introduir una espera abans d'haver creat cap tasca, ha de ser forçosament activa. No pot ser passiva, no podem canviar de tasca perquè no n'hi ha cap funcionant.

- UARTScan(): Aquesta funció l'utilitzem per llegir els missatges que rebem per la UART0, a on tenim connectada la Wifly. Ens serveix per comprovar les respostes que ens dona cada cop que fem una crida a través del driver Wifly Bàsicament el que fa és habilitar les interrupcions per aquest port. Si es produeixen són gestionades per la funció UART0_IRQHandler() de la llibreria UART, que el que fa es llegir els caràcters que es van rebent.

- parseString(): Busca si dintre d'una cadena apareix un altre cadena que indiquem. Serveix per llegir les respostes de la Wifly i veure si la comanda retorna de forma correcta, buscant per exemple la cadena “OK”.

- ➔ cutString(): Finalment amb aquesta funció retallem les respostes de la Wifly, perquè retorni només la dada que ens interessa Per exemple si volem conèixer la potència de la senyal, volem que ens retorni només un número, i no tota la cadena sencera de resposta.

Per veure més clar com funciona l'aplicació, aqui a sota podem veure el diagrama de flux:

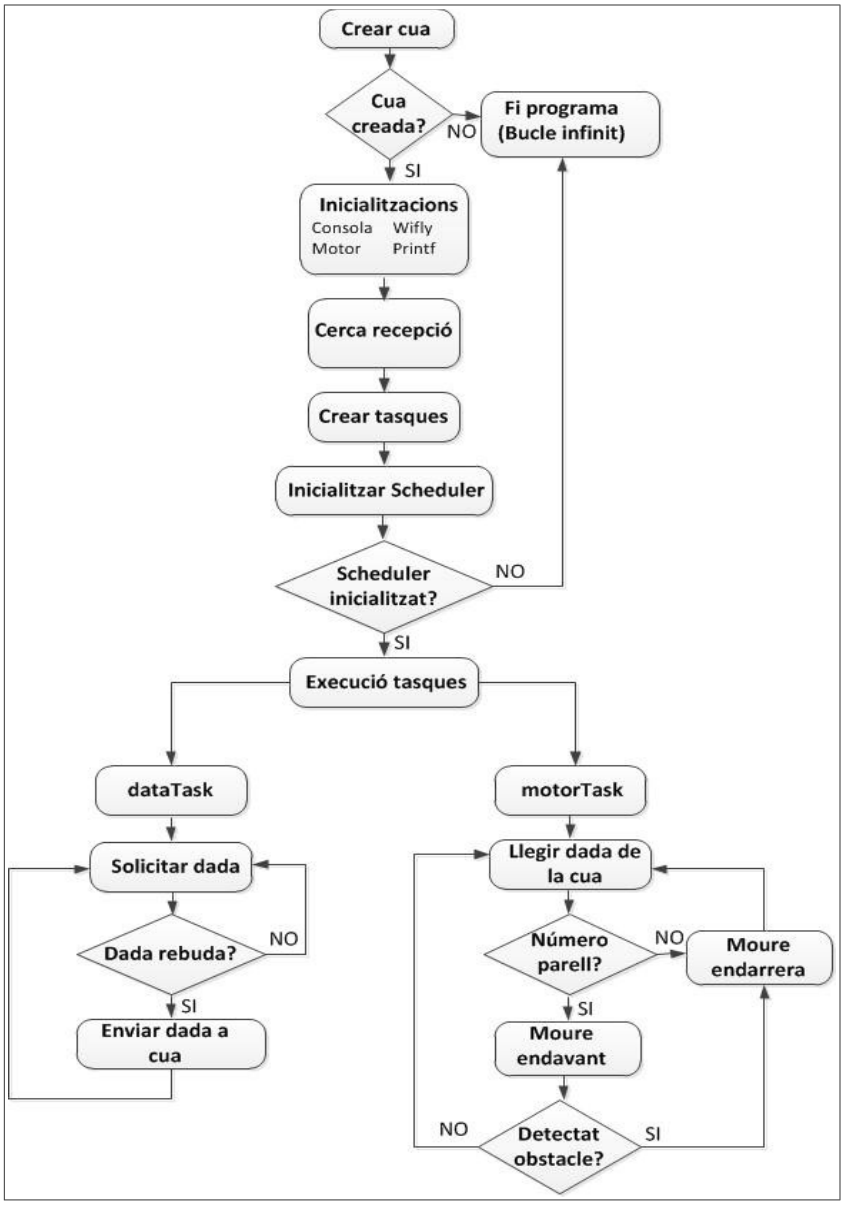
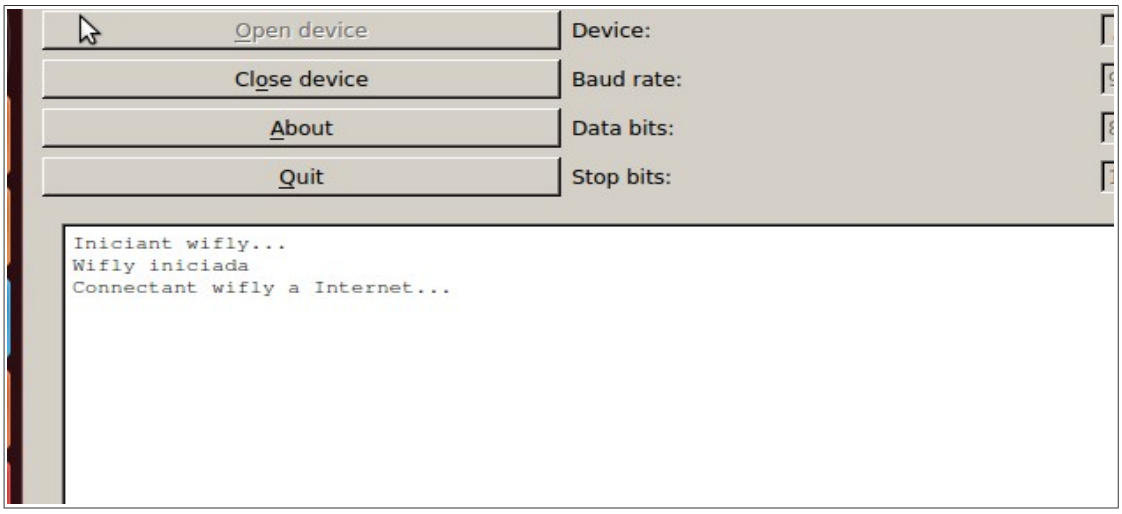
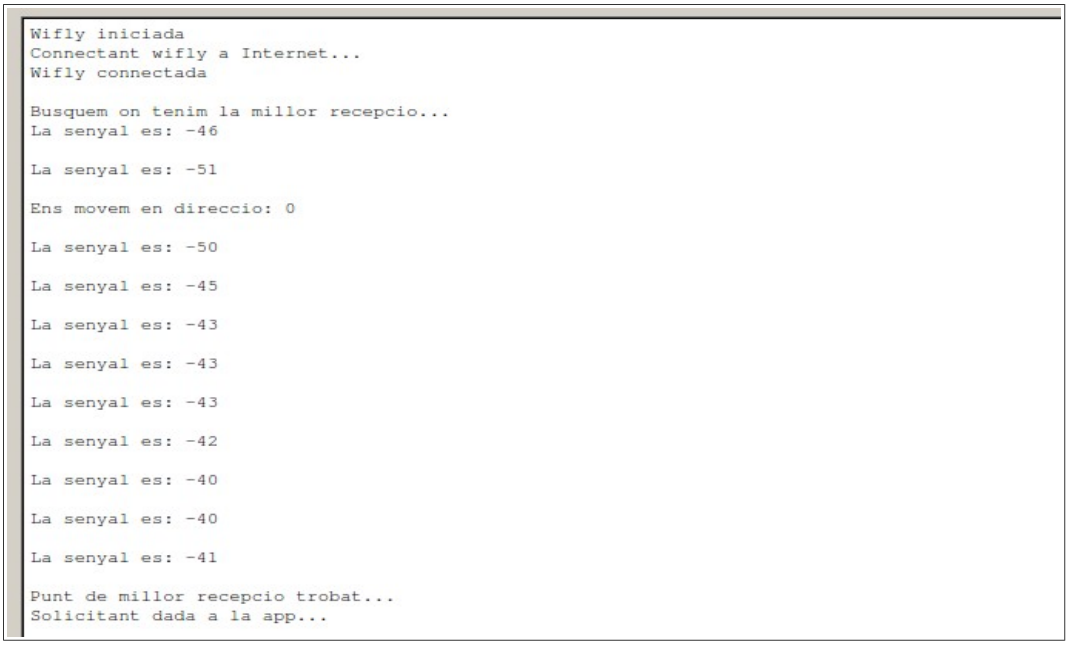


Diagrama de flux de la aplicació

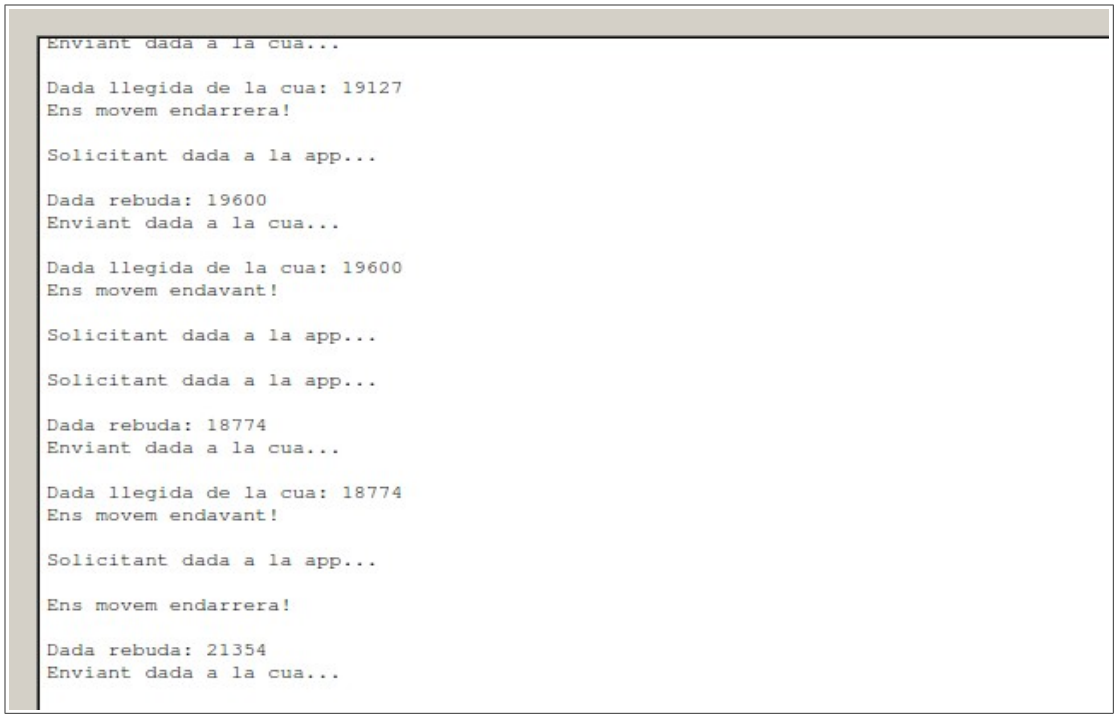
I finalment, unes captures de pantalla a on podem veure el log de les accions que va realitzant el Robot:



Captura de pantalla log 1



Captura de pantalla log 2



Captura de pantalla log 3

4.3 Sistema operatiu:

Com ja hem indicat, l'aplicació es construeix sobre un Sistema Operatiu de temps real, el FreeRTOS.

Un dels requeriments que tenen molts sistemes encastats és la possibilitat de funcionar en temps real. Hi ha algunes funcions que quan es criden, necessitem estar segurs de que s'executaran en un temps determinat, no importa el que estigui fent en aquell moment el processador. Podem pensar per exemple en el airbag d'un automòbil. En el moment en que es produeix un accident, l'airbag s'ha d'inflar en un temps concret. Si ens sortim ni que sigui uns mili-segons del temps requerit, ja serà massa tard, i no tindrà cap sentit que s'infli. Es en aquesta mena d'aplicacions que un Sistema Operatiu de temps reals mostra la seva utilitat, garantint que aquestes funcions

crítiques s'executaran sempre dintre del temps marcat.

Per poder garantir això, el FreeRTOS incorpora la possibilitat de dividir les accions a realitzar en tasques. Cada tasca porta associada una prioritat, de tal forma que s'assegura que sempre s'estarà executant la tasca amb més alta prioritat que estigui disponible. Lògicament a les funcions crítiques lis assignarem una prioritat més alta. Si varies tasques tenen la mateixa prioritat, compartiran el temps del processador de forma seqüencial.

Per gestionar això es disposa d'un planificador, en anglès *scheduler*, que controlarà quina tasca s'ha d'executar en cada moment. En el moment en que una tasca de més prioritat que la que s'està executant entri en estat de “ready”, el planificador deixarà d'executar aquesta última i executarà la de major prioritat.

Com ja hem vist, en aquest projecte es creen 2 tasques, que funcionen indefinidament. Una s'encarrega de llegir dades del servidor, i l'altre de moure el motor.

Un altre mecanisme que incorpora el FreeRTOS són les cues. Les cues són estructures tipus FIFO que es guarden en memòria. Les utilitzen les tasques per intercanviar-se missatges. Una tasca que necessiti comunicar-se amb un altre, pot enviar una dada a la cua, i l'altre tasca posteriorment la recuperarà.

En el projecte, com que la tasca que mou el motor necessita saber quina dada ha rebut l'altre tasca del servidor, es crea a l'inici una cua per poder enviar-se la dada.

Finalment, també cal destacar el mecanisme de semàfors d'exclusió o mutex. Es pot donar el cas de que 2 o més tasques d'igual prioritat necessitin accedir al mateix recurs a la vegada, la qual cosa pot crear problemes. Els mutex són com una mena de *token* necessari per poder accedir al

recurs. Quan una tasca pren el semàfor, cap altre tasca podrà accedir al recurs fins que la primera alliberi el semàfor.

En el projecte s'utilitzen diversos semàfors, per exemple en el driver de la Wifly. Si 2 tasques intenten enviar-li una comanda a la vegada, pot ser que els rebi barrejats, i per tant doni un error.

5. Viabilitat tècnica

Primer de tot cal tenir en compte que l'objectiu del projecte era bàsicament fer una petita introducció en el món dels sistemes encastats i de la programació de microcontroladors, i poder veure les múltiples possibilitats que aquests ens ofereixen. El producte final en si és bastant bàsic i no suposa cap novetat respecte al que podem trobar en el mercat. Per tant, la seva implementació només té sentit a nivell educatiu, per tots els coneixements que podem adquirir en la seva construcció.

Tècnicament, és viable implementar aquest sistema. I amb unes poques millores podríem tenir un petit cotxe teledirigit que es podria comercialitzar per entreteniment. El problema és que econòmicament no és gens viable, com veurem al següent apartat. Degut a la funció bàsicament educativa del projecte, s'utilitzen eines que estan molt per damunt de les necessitats del sistema.

Per exemple, el microcontrolador que utilitzem, el LPCXpresso1769, ens permet aplicacions bastant més complexes que aquesta. Segurament podríem haver fet el mateix amb un microcontrolador molt més senzill, d'uns pocs euros. Tampoc necessitaríem utilitzar un sistema operatiu com el FreeRTOS, ja que l'aplicació no té requeriments de temps real.

Vull remarcar finalment que en aquest apartat es parla només de la vessant tècnica del projecte. En el que es refereix a la vessant educativa, crec que el projecte té molt d'interès i compleix a la perfecció la seva funció.

6. Valoració econòmica

Per fer una valoració econòmica del projecte hem de tenir en compte, per una part, el cost dels materials, i per un altre, les hores de desenvolupament.

En quant a les hores de desenvolupament, jo hi he dedicat aproximadament 3 mesos al projecte, a una mitja d'unes 10 hores semanals. Això fan:

(90 / 7) x 10 = 128 hores.

Cal tenir en compte que moltes d'aquestes hores s'han dedicat a l'aprenentatge, i a corregir errors degut a la poca experiència. Calculo que el temps real de desenvolupament seria com a molt la meitat. Per tant, he comtat 60 hores.

Finalment, a la pàgina següent tenim la valoració econòmica total. Com es pot comprobar, la gran part dels costos són deguts a la mà d'obra. Els materials suposen una part molt petita.

PRODUCTE	PREU / UNITAT	UNITATS	TOTAL
LPCXpresso 1769	20	1	20
Mòdul RN_XV (Wifly)	40	1	40
CP2102 (UART)	4	1	4
Driver serie motor	19	1	19
Sensor contacte	5,20	1	5,20
Dual Motor GearBox	10,35	1	10,35
Toy Tires – Basic	3,80	1	3,80
Plataforma (protoboard)	6	1	6
Varis (botó, bateria, cables)	10	1	10
Mà d'obra	30	60	1800
TOTAL			**1918,35€**

Presupost

7. Conclusions

7.1. Conclusions

Recordem la llista d'objectius que ens havíem marcat a l'inici del projecte:

1.- Control del motor amb la LPCXpresso.

2.- Control del motor segons dada rebuda de l'Arpalab.

3.- Cerca de la millor cobertura.

4.- Funcionalitats addicionals.

Els 3 primers objectius formen part dels requisits del projecte. Com hem pogut veure, s'han aconseguit en la seva totalitat. El robot es mou segons les dades que va reben d'Internet. També s'ha implantat la funció de cerca de la millor cobertura, que és el primer que fa només arrancar.

Respecte al quart objectiu, està clar que és molt general, però es pot considerar que s'ha acomplert amb èxit. S'han pogut implementar 2 funcionalitats addicionals, que són un sensor per detectar obstacles i un botó de reinici del sistema. Es evident que el sistema permet afegir-hi moltes més funcionalitats, però no s'han pogut encabir dintre del marc temporal del projecte.

Per tant, com a conclusió crec que s'han acomplert els objectius marcats en aquest projecte, respectant els terminis temporals establerts.

7.2. Proposta de millores

El sistema desenvolupat en el projecte és un robot bàsic que admet moltes ampliacions i millores. Algunes possibilitats són:

- Millorar el sistema motor: Actualment el robot només es mou endavant i endarrere A més el moviment no és del tot rectilini. Seria interessant que pogués també girar cap ambdós cantons.

  El problema que ens trobem és que la controladora del motor només permet actuar sobre un motor alhora. Necessitaríem poder actuar sobre els 2 motors alhora perquè el moviment fos rectilini. Quan volguéssim girar accionaríem el motor del cantó contrari, i l'altre motor romandria parat o movent-se a una velocitat més lenta.

- Inclusió de més sensors: El robot consta d'un sensor de contacte a la part davantera. Es podrien incorporar més sensors perquè pogués interactuar molt més amb el entorn.

  Un exemple seria un sensor de proximitat. Podria ser una parella emissor-receptor de rajos infrarojos, que detectarien si tenim algun obstacle al davant i a quina distància es troba. Així podríem fer que quan estiguéssim molt aprop de l'obstacle, el robot s'aturés sense arribar a col·lisionar.

- Sistema totalment sense fils: Actualment el robot està connectat per una cable al PC. Això s'ha fet així per poder mostrar per consola un log de totes les activitats que realitza. Seria interessant poder eliminar aquest cable i que el sistema es comuniques amb l'exterior totalment sense fils.

  Una manera d'aconseguir això seria amb la inclusió d'una petita pantalla de LCD muntada

en el mateix sistema, per on podríem anar mostrant tots els missatges. Així evitaríem haver de mostrar-los per consola.

En quant a la programació de l'aplicació, es proposen les següents millores:

- Funció cerca cobertura: Durant els tests per comprovar el funcionament d'aquesta funció, s'ha detectat un “bug” que pot fer que el robot es detingui abans de aconseguir la millor cobertura

  Per trobar el punt en que la senyal és més forta, un cop s'ha detectat en quina direcció es troba el punt d'accés, es va avançant mentre la potència de la senyal vagi pujant. En el moment en que aquesta comença a disminuir, ens aturem, perquè ja hem trobat la zona de màxima cobertura. El problema és que la intensitat de la senyal no és sempre estable, pot patir pujades o baixades momentànies. Això pot ser degut a diverses raons: Interferències, persones en moviment que absorbeixen les ones, i d'altres. Una baixada momentània pot fer creure al robot que ja ha trobat la màxima cobertura, sense ser cert. Per tant en aquest cas s'aturaria abans de temps.

  Tot i que això és un factor extern i no un error de programació, es podria modificar la funció per minimitzar la possibilitat d'error. Una opció seria que, quan detectéssim una baixada de la senyal, ens aturéssim i prenguéssim varies mesures a intervals definits. Després, hauríem d'utilitzar algun algorisme per decidir quina és la mesura correcte (la més alta, una mitja aritmètica, etc). Cal tenir en compte, però, que això allargaria la duració del procés.

- Detecció de col·lisió: El sistema com hem dit incorpora un sensor de contacte, que detecta quan s'ha produït una col·lisió. Cada cop que hem d'avançar, al finalitzar el

moviment comprovem la senyal provinent del sensor, i si s'ha detectat col·lisió tornem cap enrrera.

Una millora interessant seria que el sensor ens pogués comunicar exactament el moment en que es produeix la col·lisió, així podríem aturar el moviment de seguida, sense esperar a finalitzar-lo. Això ho podríem fer mitjançant l'us d'interrupcions. Cada cop que el nivell de la senyal d'entrada del pin del sensor passés a ser alt (contacte detectat), el microcontrolador rebria una interrupció i podria detenir el moviment.

7.3. Autoavaluació

Vaig escollir pel treball l'àrea de “Sistemes Encastats” perquè, tot i no tenir gaire coneixement sobre el tema, em va semblar molts interessant. Es un àrea que agrupa diferents disciplines, i que m'ha permès aplicar coneixements apresos en diverses assignatures de la carrera: electrònica, sistemes operatius, programació, fonaments de computadors i xarxes de comunicacions.

La veritat és que la corba d'aprenentatge ha resultat al inici bastant elevada. En alguns d'aquests camps, per exemple la programació, jo tenia molt poca experiència. En certs moments l'experiència ha estat inclús un pel frustrant. Però amb l'esforç, la consulta en diferents fonts, i l'ajuda del consultor i dels companys de l'aula, m'ha permès poc a poc anar superant tots els obstacles.

Crec que amb la realització del projecte he aprés molt. Si miro 3 mesos endarrere veig que he fet un gran avanç en el coneixement dels microcontroladors i dels Sistemes encastats. Encara em queda molt més per aprendre, però crec que he pogut assentar una bona base

Com a resum, puc dir que ha estat una experiència apassionant i molt profitosa.

8. Glossari

Diagrama de Gantt: Eina gràfica utilitzada per planificació de treballs i projectes, a on es representen totes les tasques a realitzar i el temps en que s'han de realitzar.

FIFO (First in, first out): Es un sistema d'ordenació d'una cua en el qual l'element que primer hi entra és el primer que hi surt.

FreeRTOS: Sistema Operatiu en temps real sobre el qual s'ha programat l'aplicació del projecte.

GPIO (General purpose input/output): Pins en el microcontrolador que poden ser programats per ser utilitzats com a entrada o sortida digital.

JTAG: Es una especificació per depurar en temps real i sobre la mateixa plataforma que s'utilitza en una gran quantitat de dispositius electrònics.

Microcontrolador: Circuit integrat programable, que inclou en una sola placa les 3 unitats funcionals d'una computadora: Unitat central de procés, memòria i mòduls de E/S.

Scheduler (planificador): Entitat dintre d'un sistema operatiu que s'encarrega de gestionar l'execució de les diferents tasques.

Sistemes encastats: Sistema informàtic que es troba inclòs dintre d'un altre sistema, i que normalment es programa per una tasca única i no pot ser reprogramat per l'usuari.

Sistema operatiu de temps real: S'utilitzen per crear aplicacions en temps real, que són aplicacions que inclouen accions crítiques que hem de garantir que s'executaran sempre dintre d'un temps limit definit.

UART: Sigles en anglès de “Transmissor / receptor asíncron universal. Es un petit circuit integrat utilitzat per a comunicacions sèrie, Transforma dades entre sèrie i paral·lel.

9. Bibliografia

Llibres

- **Brian W. Kernighan i Dennis M. Ritchie**, “El Lenguaje de programación C”, segunda edición (2011), Editorial Prentice-Hall.
- **Richard Barry**, “Using the FreeRTOS real time kernel, LPC17xx Edition”, Real Time Engineers Ltd.
- **José María Gómez Cama i altres**, “Sistemes encastats”, (2011). material docent de la UOC.

Internet

- Wiki de l'assignatura, http://cv.uoc.edu/app/mediawiki14/wiki/IniciCortexM3
- Web del FreeRTOS, http://www.freertos.org/
- Web de Code Red Technologies Ltd, http://www.code-red-tech.com/
- Viquipèdia, http://ca.wikipedia.org/wiki/Portada

Documents PDF

- Datasheet LCPXpresso1769, “ lpc1769.pdf ”.
- Guia usuari LPCXpresso1769, “ lpcxpresso.getting.started.pdf ”.
- Manual usuari LPCXpresso1769, “ UM10360.pdf “.
- Datasheet Wifly, “ WiFly-RN-XV-DS.pdf “.
- Manual usuari Wifly, “ WiFly-RN-UM.pdf “.

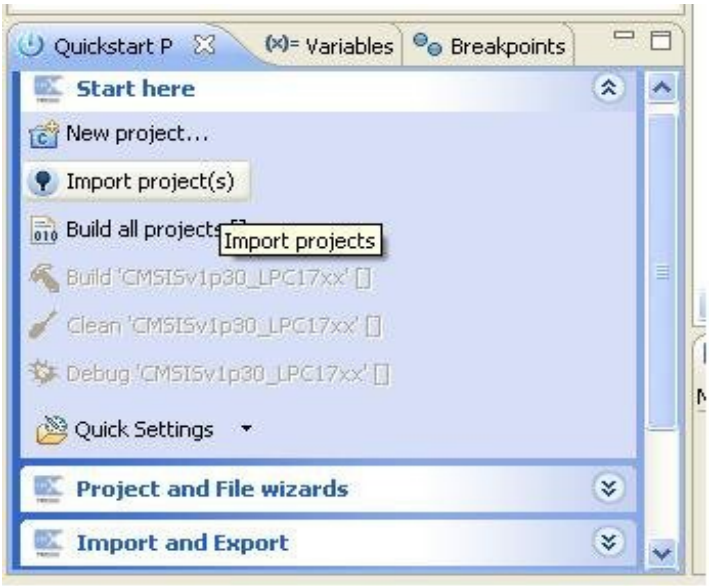
10. Annexe

10.1. Execució i compilació

L'entrega final consisteix en un arxiu comprimit que conté un “workspace” del IDE del LCPXpresso. Recordem que aquesta plataforma inclou un entorn de desenvolupament basat en “Eclipse”. Dintre del “workspace” s'inclouen diversos projectes. L'aplicació principal es troba en el “RobotSimple”. La resta de projectes són llibreries de suport.

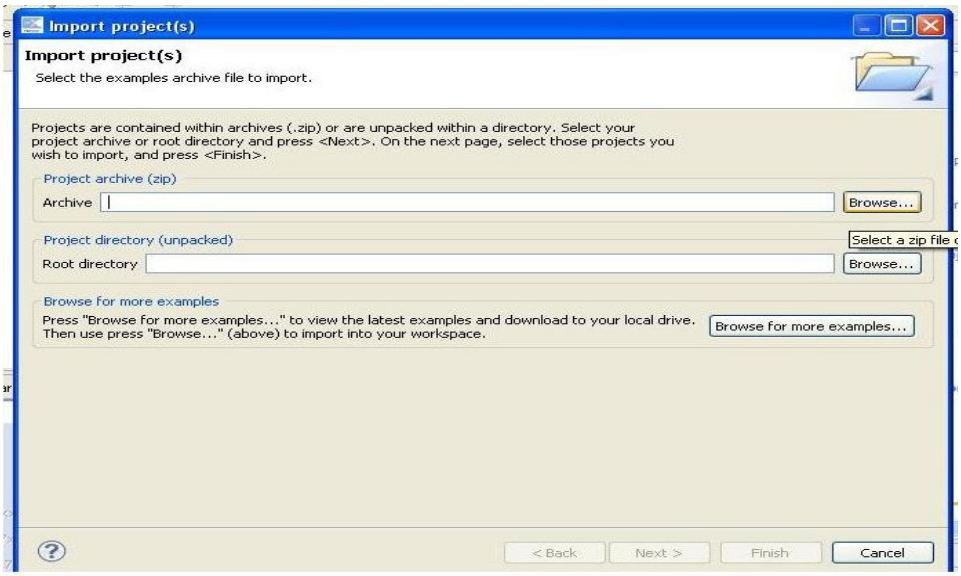
Els passos per poder compilar i executar l'aplicació són els següents:

Primer de tot hem d'importar el projecte al “workspace” actual. Això ho fem amb l'opció “import project”, al “quickstar panel”, situat al cantó inferior esquerra si utilitzem la configuració del espai de treball per defecte.



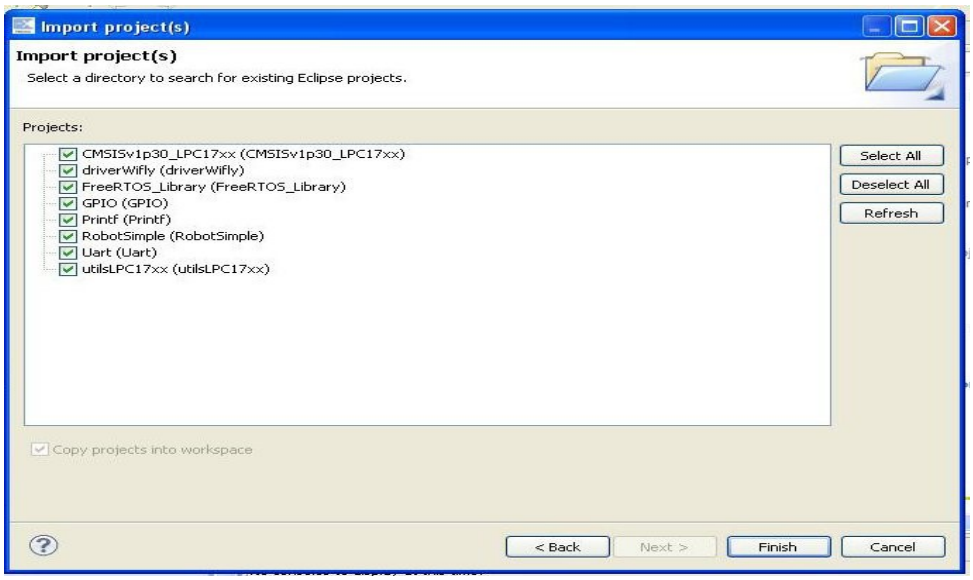
Pantalla importar projecte

Seleccionem l'arxiu comprimit que s'ha entregat:



Pantalla selecció arxiu projecte

I seleccionem tots els projectes per incorporar al “workspace”:



Pantalla selecció projectes

Un cop tenim els projectes en l'espai de treball, hem de compilar-los. Per fer-ho només cal pitjar el botó “Build all projects”, també al “quickstart panel”.

Finalment ens queda només carregar l'aplicació a la memòria flash de la LPCXpresso. Primer de tot, ens assegurarem que aquesta està connectada a un dels ports USB de l'ordinador. Després, si tenim seleccionat el projecte “RobotSimple”, en el mateix “quickstart panel” tenim l'opció “Debugg RobotSimple”, que ens permet *debuggar* i també carregar la flash al microcontrolador Un altre opció seria pitjar a la barra de menús “Program flash”, i al quadre de diàleg seleccionaríem l'arxiu amb extensió “axf”, dintre de la carpeta debug al projecte principal.

Un cop tenim la LPCXpresso programada, ja la podem desconnectar del PC, i reiniciar-la perquè comenci a executar el programa.