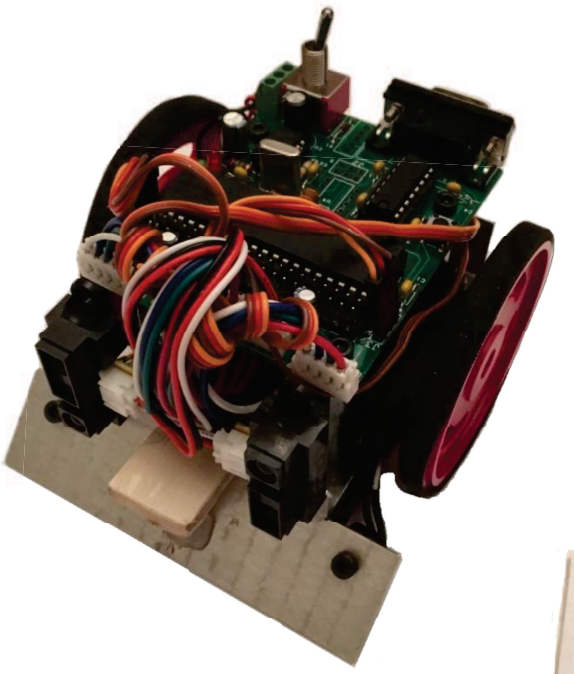
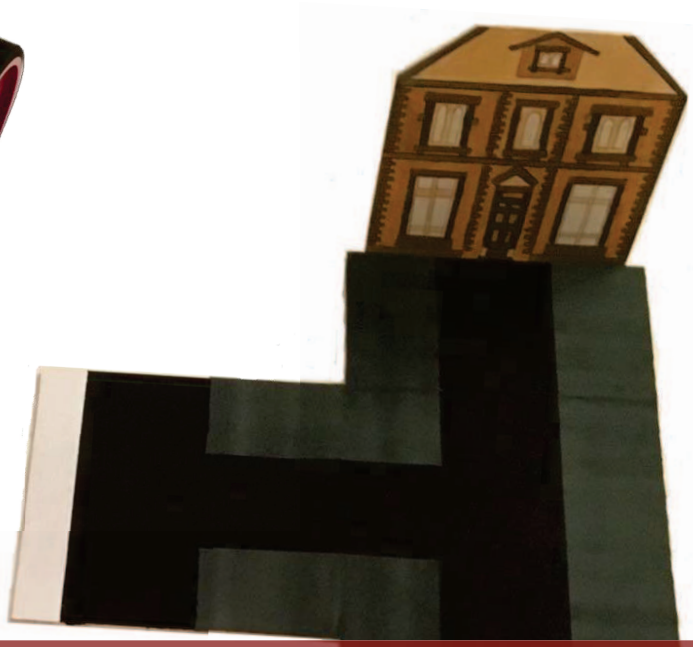
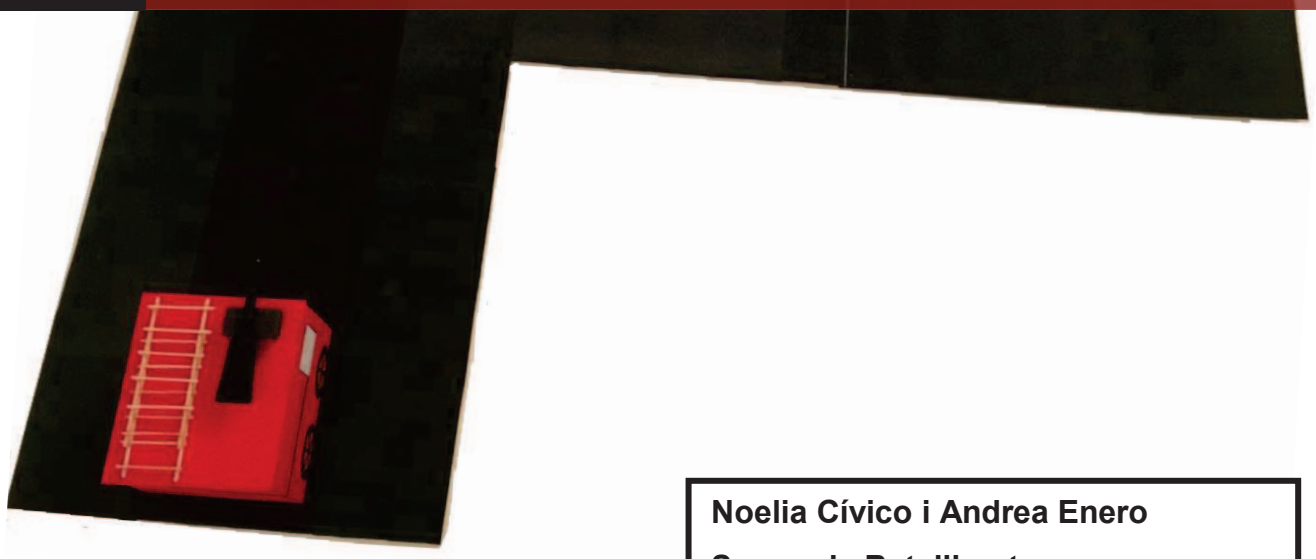
“At bottom, robotics is about us. It is the discipline of emulating our lives, of wondering how we work.”

Roderic Grupen (1961, USA)





DISSENY I PROGRAMACIÓ D'UN COTXE DE BOMBERS



Noelia Cívico i Andrea Enero

Segon de Batxillerat

INS Les Marines

26 de Gener del 2016

Tutoritzat per Pilar Ruiz Bellido

ÍNDEX DE CONTINGUTS

1. Introducció i objectius

Aquest treball de recerca consisteix en la construcció d'un robot seguidor de línia i amb sensors de distància.

La idea de la construcció del robot prové de l'oferta de la professora de tecnologia per formar part d'un curset de programació i robòtica que s'impartia al llarg de tres setmanes durant el mes de juliol del 2015 a la Universitat Politècnica de Catalunya. Aquest curset es basava en l'aprenentatge de coneixements bàsics entorn la programació en C (tipus de llenguatge emprat en el disseny de diferents programes) i, seguidament, la creació d'un robot base amb el primer objectiu de formar part d'una competició de robots jugadors de sumo.

Per tant, fins al moment la hipòtesi del treball era:

Es pot aconseguir construir un robot seguidor de línies i capaç de guanyar una competició de sumo contra altres jugadors?

Al veure que es va superar amb èxit i sense grans dificultats el primer objectiu, vam decidir ampliar la hipòtesi inicial i ens vam plantejar la següent hipòtesi final:

Es pot aconseguir construir un robot caracteritzat com si fos un cotxe de bombers i capaç de seguir un circuit que simuli el nucli urbà d'una ciutat fictícia?

Aquesta segona idea va sorgir per la intenció de crear un robot automòbil més complex. Vam concloure que un cotxe de bombers s'ajustaria a aquestes condicions; ja que alguns components del vehicle serien fàcilment introduïbles en el robot dissenyat. Per exemple, en un primer moment, vam pensar en la introducció d'una sirena, una petita bomba d'aigua o unes llums d'alerta.

Per tant, va ser llavors quan vam iniciar la realització del projecte final; és a dir, el disseny del cotxe de bombers en una ciutat fictícia.

Pel que fa als objectius, algunes propostes principals en aquest treball de recerca van ser:

- Aprendre i dominar la programació en C.
- Millorar les nostres habilitats a l'hora de soldar components electrònics.
- Programar de manera adequada el robot perquè romangui durant el transcurs de la lluita dins del ring circular i sigui capaç de atacar al contrincant.
- Dissenyar de manera creativa la nostra maqueta.
- Programar el robot perquè segueixi el circuit establert en la maqueta i superi els diferents obstacles.
- Treballar de manera cooperativa amb la companya.
- Aconseguir fer una demostració perfecta del funcionament del robot el dia de la presentació del treball de recerca.

2. Curset d'Iniciació a la programació i a la robòtica

2.1. Introducció teòrica (primera setmana)

Al llarg de la primera setmana de curset, vam estudiar els següents conceptes: sistema binari, sistema hexadecimal i programació en C.

2.1.1. Sistema binari

Aquest tipus de sistema consisteix en escriure els números decimals com combinacions de 1 i 0. Això s'aconsegueix dividint el número decimal entre 2 i agafant l'últim quocient i tots els residus en ordre ascendent.

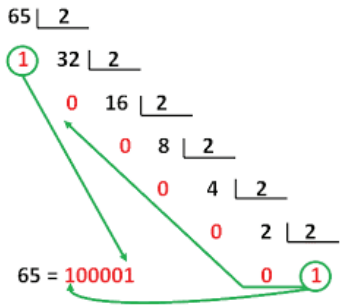


Figura 1. De decimal a binari (Font d'Internet).

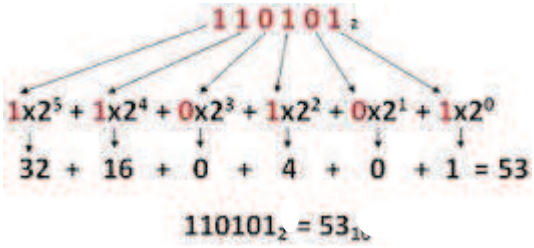


Figura 2. De binari a decimal (Font d'Internet).

2.1.2. Sistema hexadecimal

Aquest tipus de sistema consisteix en representar els números de manera simplificada, utilitzant els 10 primers dígits de la numeració decimal i simbolitzant els números del 10 al 15 amb les sis primeres lletres de l'alfabet.

DECIMAL	HEXADECIMAL	BINARI	DECIMAL	HEXADECIMAL	BINARI
0	0	0000	8	8	1000
1	1	0001	9	9	1001
2	2	0010	10	A	1010
3	3	0011	11	B	1011
4	4	0100	12	C	1100
5	5	0101	13	D	1101
6	6	0110	14	E	1110
7	7	0111	15	F	1111

2.1.3. Programació en C

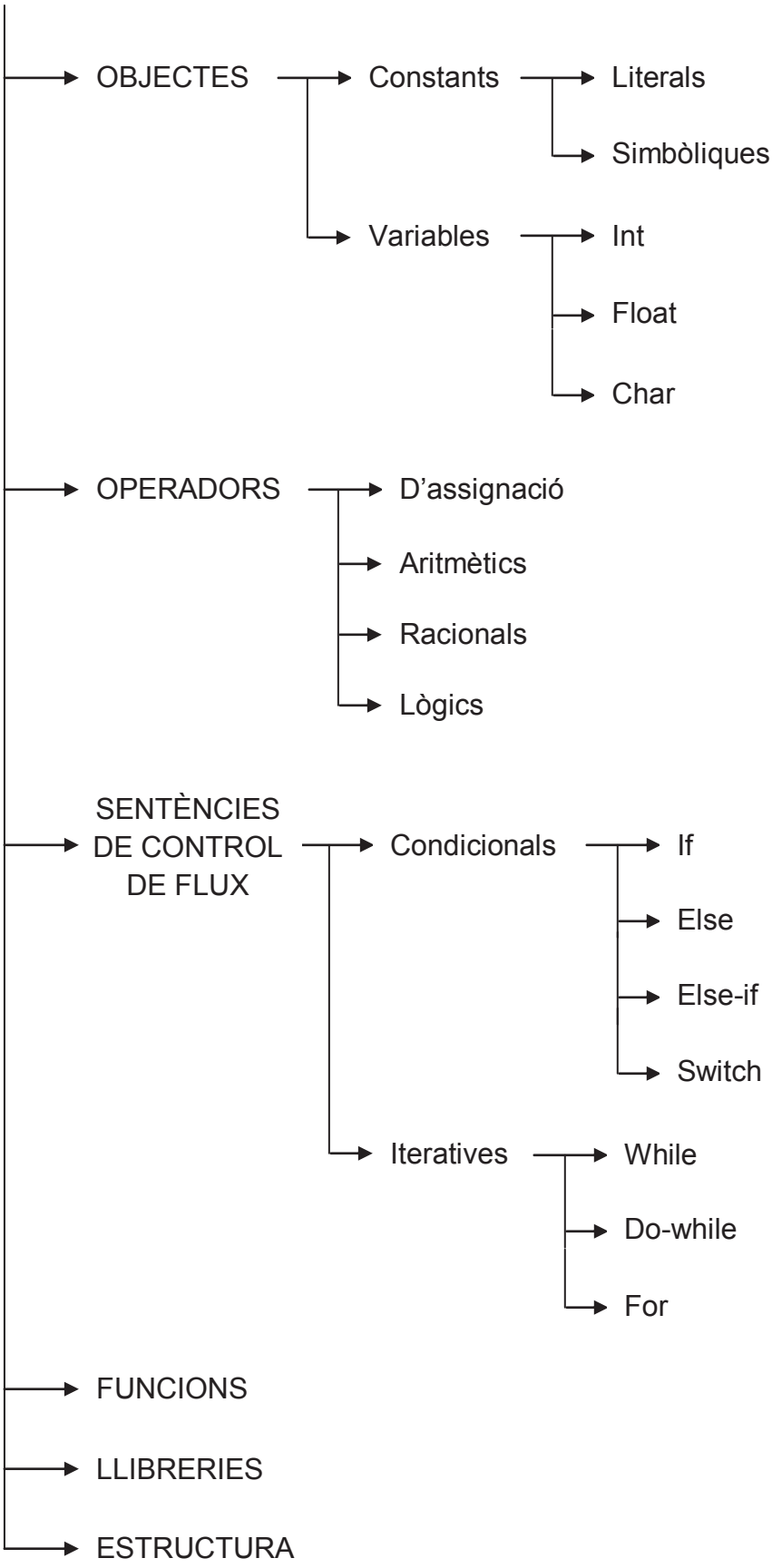
Per controlar el moviment d'una màquina, s'utilitza un llenguatge de programació que permet especificar tant la classe de dades amb què treballarà com les accions que realitzarà.

El llenguatge de programació en C va ser desenvolupat per Dennis Ritchie en els Laboratoris Bell de l'empresa de comunicacions AT&T en 1972. Aquest tipus de programació va ser creada inicialment amb un propòsit molt concret: el disseny d'un sistema operatiu. No obstant això, aviat es va revelar com un llenguatge molt potent i flexible, fet que va provocar que el seu ús s'estengués ràpidament. D'aquesta manera, programadors de tot el món van començar a fer servir el llenguatge en C per escriure programes de tot tipus.

El llenguatge en C és el que vam estudiar en el curset; donat que només són necessàries unes poques instruccions per a que la màquina sigui capaç de traduir cada element del llenguatge, sense que calgui un suport intens en temps d'execució.

En la següent pàgina, podem trobar un esquema del funcionament del llenguatge en C, on es pot identificar tots els elements que el formen i les seves diferents estructures; i, a continuació, una explicació detallada de les idees principals d'aquest tipus de llenguatge.

PROGRAMACIÓ EN C



OBJECTES

En la programació en C s'utilitzen diferents objectes, que són els mecanismes que emmagatzemen el conjunt de valors necessaris en els diferents programes d'ordinador.

Per una banda, existeixen les **constants**, que tenen un valor fix que no pot ser modificat. Aquestes poden ser de dos tipus:

- Constants **literals**: el seu valor apareix directament en el codi cada vegada que és necessari per a una operació.
- Constants **simbòliques**: són representades mitjançant un nom (símbol) en el programa. Per utilitzar el seu valor constant, s'utilitza el seu nom simbòlic, de la mateixa manera que ho faríem amb una variable. A continuació, es mostra un exemple d'aquest tipus de constant:

Per calcular el perímetre d'una rodona, creem la següent sentència:

*Perímetre= 2*3,14*radi;*

Per evitar posar en cada sentència el valor del nombre Pi, podem crear una constant:

Definim PI:3,14;

Aleshores, la sentència inicial es veuria simplificada:

*Perímetre= 2*PI*radi;*

L'estructura de qualsevol constant és la següent:

```
#define nom_constant valor_constant
```

Per una altra banda, les **variables** són els objectes on es guarden uns valors, els quals poden ser consultats i modificats durant l'execució del programa.

Tota variable s'ha de declarar abans de ser usada per primera vegada en el programa. Les sentències de declaració de variables indiquen al compilador que ha de reservar cert espai a la memòria del ordinador per tal d'emmagatzemar una dada. Hi ha tres tipus de variables i, depenent d'això, es reserva més o menys espai.

Les variables poden ser:

- Tipus **int**: per a valors enters (exemple: 3)
- Tipus **float**: per a valors reals (exemple: 3,45)
- Tipus **char**: per declarar caràcters (exemple: a)

En les variables s'ha de tenir en compte no perdre informació; ja que si tenim una variable de tipus *int* i la guardem en una altra de tipus *float*, el nostre valor segueix sent el mateix, però, en el cas contrari (en passar de *float* a *int*) es produeix una pèrdua d'informació.

L'estructura de les variables és la següent:

- tipus nom_variable;
  Per a declarar una variable d'un tipus.
- tipus nom_variable1, nom_variable2;
  Per a declarar més d'una variable d'un mateix tipus.
- tipus nom_variable = valor_variable;
  Per a donar un valor a una variable.

OPERADORS

A més dels nombrosos objectes, s'han de tenir en compte els diversos **operadors**, que n'hi ha de quatre tipus:

- Operadors **d'assignació**: Ens permeten assignar valors a les variables. El seu símbol és el signe =. Aquest operador assigna a la variable que està a la seva esquerra el valor que es troba a la dreta. Seguidament, s'exemplifica aquest tipus d'operador:

```
x=1
```

- Operadors **aritmètics**: Ens permeten operar amb les constants i les variables. En la següent taula es troben tots el operadors aritmètics que s'usen en la programació en C, el seu signe i la seva funció.

UNARIS	Signe negatiu	-	El negatiu de a serà -a
	Increment	++	a++ = a+1
	Decrement	--	a-- = a-1
BINARIS	Suma	+	a+b = c
	Resta	-	a-b = c
	Multiplicació	*	a*b = c
	Divisió	/	a/b = c
	Mòdul	%	a%b = c (on c és el residu de la divisió)

- Operadors **racionals**: S'utilitzen per crear condicions en sentències condicionals o iteratives, que veurem més endavant. Els operadors racionals utilitzats en la programació en C són els següents:

Menor que	<
Major que	>
Menor o igual que	<=
Major o igual que	>=
Igual que	==
Diferent de	!=

Al relacionar o comparar dues expressions mitjançant un d'aquests operadors, s'obté un resultat lògic; és a dir: "cert" o "fals". Per exemple, la sentència 8>4 dóna com a resultat el valor "cert"; en canvi, la sentència 7<=2 dóna com a resultat el valor "fals".

- Operadors **lògics**: S'utilitzen per elaborar condicions complexes. Els diferents tipus d'operadors lògics són:

AND (multiplicació)	&&
OR (suma)	||
NOT (negació)	!

Per millor el funcionament del llenguatge en C, es va declarar una prioritat entre tots el operadors. Així doncs, en una mateixa sentència és pot fer servir més d'un operador i el programa seguirà el següent ordre:

1) Parèntesis ()
2) Negat (!)
3) Operadors aritmètics (- , ++, --, *, /, %, +, -)
4) Operadors racional (<=, >, >=, ==, !=)
5) Operadors lògics (&&, ||)
6) Assignacions (=)

SENTÈNCIES DE CONTROL DE FLUX

Un dels aspectes més importants i representatius del llenguatge en C són les **sentències de control de flux**, que s'usen als programes per canviar el flux seqüencial de l'execució. Hi ha dos tipus de sentències de control de flux: les sentències **condicionals** i les sentències **iteratives**.

Les sentències condicionals són estructures que permeten que una sentència o un grup de sentències s'executin si es dóna una condició; és a dir, executar una part o una altra de codi condicionalment.

Les diferents maneres que el llenguatge en C ofereix per controlar el flux d'execució d'un programa de forma condicional són:

- Construcció **if**: S'executen un conjunt de sentències només si la condició és certa; i, per crear aquesta condició, s'utilitzen els operadors i les variables. L'estructura d'aquest tipus de sentència es la següent:

```
if (condició) {
acció;
 }
```

Això significa que si la condició és certa s'executarà l'acció que hi ha dins dels claudàtors.

- Construcció **else**: En aquest cas, quan la condició de l'*if* és falsa, es fa l'acció que hi ha dins de l'*else*. Per tant, per a que hi hagi una condició *else*, ha d'haver una condició *if* principal. L'estructura d'aquesta construcció és la següent:

```
if (condició) {
 acció1;
}
else {
acció2;
}
```

En aquest cas, si la condició és certa, el programa executarà l'acció 1; en canvi, si la condició no es compleix, l'acció 2 serà l'executada.

- Construcció **else-if**: Aquesta és una construcció condicional alternativa als *if*. Es basa en una sèrie de condicions plantejades de manera esglaonada. Seguidament, es troba la seva estructura:

```
if (condició 1) {
acció1;
} else if (condició 2) {
acció2;
}
```

Per tant, si la condició del primer *if* és certa, s'executarà l'acció 1. Per altra banda, si és falsa, el programa comprovarà si la condició 2 és certa. Si és així, s'executarà l'acció 2. En el cas que sigui falsa, no es realitzarà cap acció.

- Construcció **switch**: És una sentència condicional que depèn del valor d'una variable, la qual permet executar una o varies sentències d'entre moltes. Aquesta construcció compara la variable que ens interessa amb els diferents casos que indiquem (als quals anomenem *case*). La seva estructura és la següent:

```
switch (expressió) {
case valor1:
sentencia1;
break;
case valor2:
sentencia2;
break;
}
```

En primer lloc, s'avalua l'expressió. Seguidament, el seu valor es comparat seqüencialment amb els valors dels diferents *case* (el nombre de *case* per cada switch no està restringit). Si el valor de l'expressió coincideix amb algun valor dels diferents *case*, s'executaran les sentències corresponents fins que s'arribi a un *break*, el qual tancarà la construcció *switch*.

Per un altre costat, les sentències iteratives són un conjunt que executa un mateix grup de sentències durant un seguit nombre de vegades. Així doncs, podem dir que són mecanismes per a la creació de bucles dins d'un programa.

En el llenguatge en C trobem els següents tres tipus de sentències iteratives:

- Construcció **while**: Aquesta sentència repeteix un conjunt de sentències sempre que es compleixi una condició inicial. El seu funcionament és bastant simple: el cos del bucle (és a dir, la sentència o grup de sentències dins del bucle) s'executa mentre el valor de la condició sigui cert. En el moment en que la condició sigui falsa, l'execució del programa continua seqüencialment amb la següent instrucció després del bucle. La seva estructura és la següent:

```
while (condició) {
sentencies;
}
```

- Construcció **do-while**: Aquesta construcció funciona de manera molt similar a la construcció *while*. No obstant això, al contrari que aquesta, *do-while* executa primer el cos del bucle i després avalua la condició; per la qual cosa, el cos del bucle s'executa com a mínim un cop. Seguidament, trobem la seva estructura:

```
do {
sentencies;
} while (<condició>)
```

Tal com s'ha dit, inicialment s'executaran les sentències que es troben dins del *do*. Seguidament, es comprovarà si la condició del *while* és certa o falsa. En el primer cas, es tornarà a repetir tot el procés. En l'altre cas, el programa s'acabarà.

- Construcció **for**: És una construcció més fàcil d'explicar a partir de la seva estructura:

```
for (sentencia inicial ; condició ; increment/decrement){
grup de sentencies;
}
```

La primera part de la construcció *for* acostuma a ser una sentència inicial d'assignació, on es declara alguna variable que controla el número de vegades que s'ha d'executar el cos del bucle. Aquesta sentència s'executa en una sola ocasió: abans d'entrar per primera vegada al cos del bucle. En segon lloc, trobem una condició: en el cas de que sigui certa, es seguirà amb el programa. Tercerament, trobem una sentència d'increment o de decrement que afecta a la variable declarada en la sentència inicial. Aquesta tercera part s'executa sempre després de l'execució del cos del bucle. Finalment, trobem el grup de sentències que s'executen dins del *for*. Al ser una construcció iterativa, seguirà en un bucle fins al moment en que la condició sigui falsa; en aquest cas, el *for* s'acabarà.

FUNCIONES I LLIBRERIES

A més, dos elements necessaris per la programació en C són: les **funcions** i les **llibreries**.

Les funcions permeten modular un programa i fan que el seu desenvolupament sigui més fàcil de dur a terme. Cadascuna de les funcions es limita a executar una senzilla funció, que estarà definida en el seu nom. L'estructura per definir una funció és la següent:

```
tipus nom_funció (...variables...){
sentencies;
}
```

Les variables utilitzades poden ser tant locals com globals.

Durant el programa, per utilitzar una de les funcions definides, només fa falta cridar-la. Aquesta acció es realitza de la manera que s'indica a continuació:

```
nom_funció (paràmetres);
```

Per un altre costat, les llibreries són recopilacions de fitxers, preestablertes o creades per un mateix, que implementen operacions comunes, com les d'entrada o de sortida. Tenen la següent estructura:

```
#include <nom_llibreria.h>;
```

ESTRUCTURA

Per últim, l'**estructura** és un punt indispensable. Les parts que ha de tenir qualsevol programa són, de manera ordenada:

1) Declaració de llibreries

2) Definició de constants

3) Capçaleres i estructures de les funcions que utilitzarem

4) Definició de variables globals

5) Programa

A més, també és necessari saber que existeix la possibilitat d'afegir comentaris o notes entremig del programa. Simplement, s'ha d'introduir al començament el següent signe: //.

2.2. Introducció pràctica (segona setmana)

Al llarg dels següents cinc dies del curset de programació i robòtica, ens vam traslladar a l'aula d'ordinadors, on hi ha instal·lats també els multímetres.

Un cop allà, vam començar a posar en pràctica els coneixements adquirits; aprenent a manejar el programa de programació de sistemes i el simulador.

Les diferents activitats realitzades van ser: simulació d'un comptador d'una xifra, simulació d'un comptador de dues xifres, el cotxe fantàstic, display d'un comptador d'una i dues xifres, i display d'un comptador de segons d'una i dues xifres.

En primer lloc, la simulació d'un comptador d'una xifra i el de dues consistia simplement en realitzar el programa, on s'establia un bucle que sumés un número cada vegada (primer només per la xifra de les unitats i després per la de les unitats i també les desenes). Després, es carregava el programa en el simulador i s'observava el seu funcionament de manera virtual.

En el primer cas, el programa consisteix bàsicament en un *while* on la condició és sempre veritat i, per tant, s'estableix un bucle infinit i la sentència es repetirà contínuament. A més, també podem trobar una sentència iterativa *for*; on declarem una variable, que és el número, una condició, que és que aquest no pot superar el 10 i, finalment, una sentència d'increment que augmenta una xifra la variable del número. Dins de les sentències trobem l'ordre que dictamina l'encesa de les diferents parts del display per així poder formar els números. Aquestes diferents parts que s'han d'il·luminar venen declarades en una llista amb el conjunt de nombres unitaris.

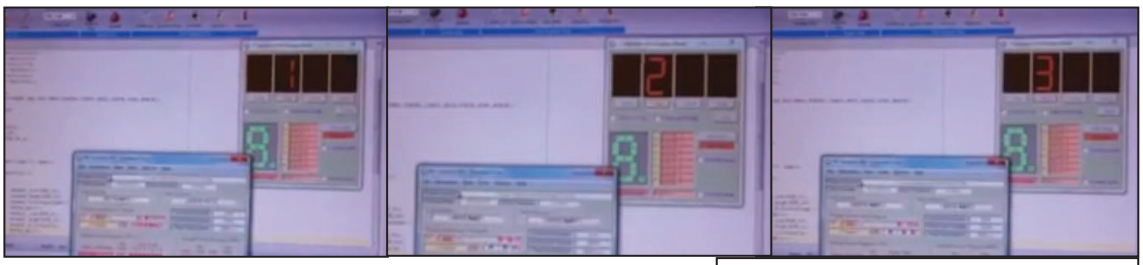
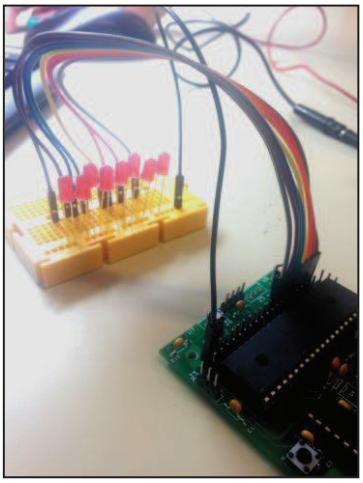


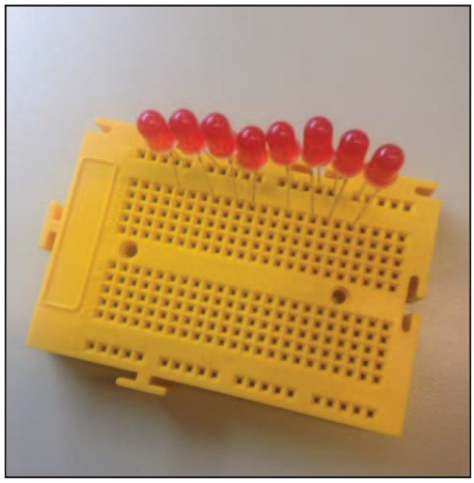
Figura 3. Imatge del comptador d'una xifra (Font pròpia).

Segonament, el programa de comptador de desenes té la mateixa estructura que l'anterior; però en aquest cas hi trobem dos displays i, per tant, el programa està adaptat per a que el segon display vagi augmentant de nombres fins arribar al 9, que serà quan el primer display augmenti una xifra, formant d'aquest manera tots els números entre el 01 fins al 99.

En segon lloc, el cotxe fantàstic era una pràctica que es basava en un conjunt de vuit leds, posicionats sobre la *protoboard*, o placa de proves, que estava connectada a la placa i la placa a la vegada estava unida al multímetre. D'aquesta manera, es carregava el programa dins de la placa; i això feia que els leds s'encenguessin d'un en un cap endavant i cap endarrere intermitentment.

En aquest programa vam tornar a declarar un bucle infinit per mitjà d'un *while*, i vam fer servir les condicions *output_high* i *output_low* per a que els leds s'encenguessin i apaguessin, intercalades amb una *delay* o retard, aconseguint així que els leds es mantinguin encesos al llarg d'una petita fracció de temps.

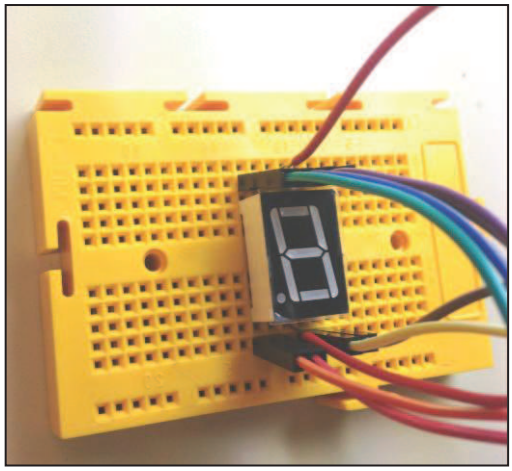


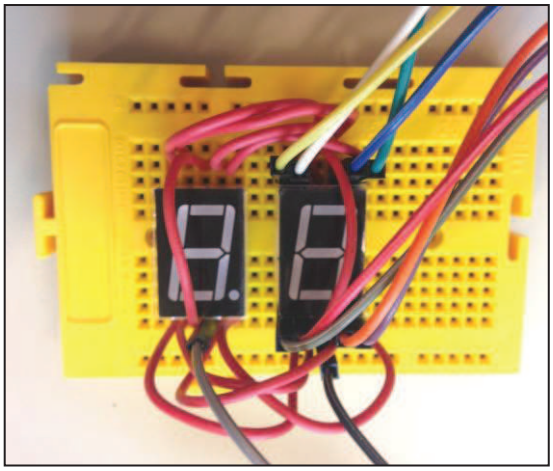


Figures 4 i 5. Imatges de la protoboard i de les connexions amb la placa (Font pròpia).

En tercer lloc, el display d'un comptador d'unitats i de desenes incorporava un i dos displays, respectivament; situats sobre la *protoboard*. En aquest cas, el mateix programa del comptador d'unitat i de desenes es carregava en la placa construïda prèviament (explicació en el següent apartat). Per poder dur-ho a

terme, s'havien de realitzar també les diferents connexions: primer de tot, la connexió de la placa amb el multímetre; i, a continuació, la connexió de la placa amb la *protoboard*.





Figures 6 i 7. Imatges del comptador d'una i de dues xifres en la protoboard (Font pròpia).

Per últim, el display d'un comptador de segons d'una i de dues xifres era pràcticament el mateix que el display d'un comptador normal; només s'havia d'afegir una funció per controlar el pas dels números a temps real.

En l'apartat d'annexos, estan adjuntats tots aquests programes (8.1, 8.2, 8.3, 8.4, 8.5, respectivament). A més, en l'USB entregat conjuntament amb el treball, es troba una demostració en vídeo d'aquestes pràctiques.

2.3. Programació i construcció del robot (tercera setmana)

L'última setmana consistia bàsicament en la construcció i la programació del robot, que era la nostra primera hipòtesi.

Primerament, l'objectiu era soldar amb estany a la placa la següent llista d'elements:

1. Resistències
2. Diode
3. Condensador ceràmic
4. Polsador RESET
5. Sòcols
6. Leds
7. Tires de pins
8. Regleta
9. Condensadors electrolítics
10. DB9
11. Clock
12. Interruptor
13. Connectors blancs (després de comprovar la placa i fer les diferents pràctiques)

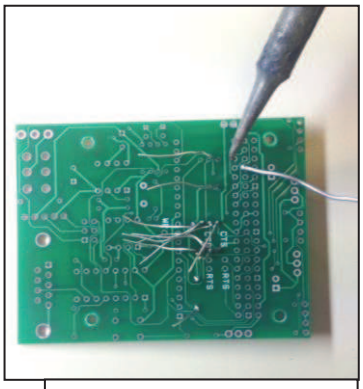


Figura 8. Soldadura de la placa (Font pròpia).

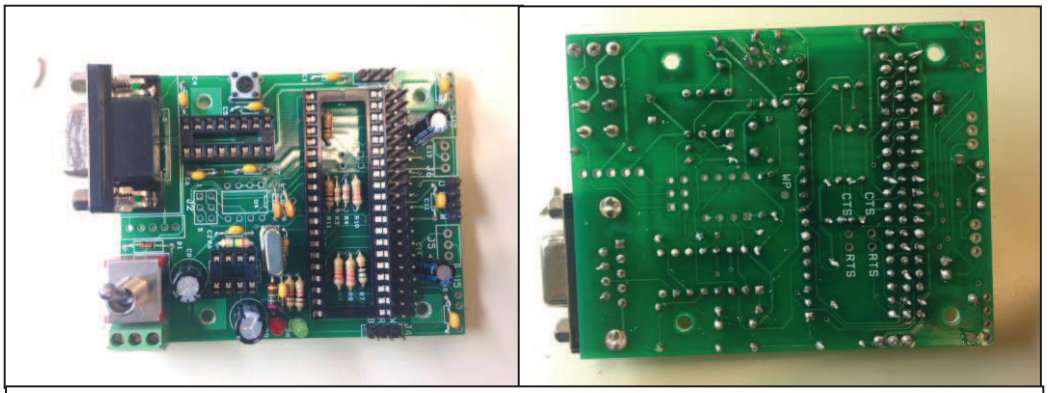


Figura 9. Placa amb tots els components soldats (Font pròpia).

Tot seguit, aquesta s'havia de comprovar; donat que és l'element essencial del robot, la responsable del seu funcionament i on es carrega el programa. A més i abans de seguir amb la construcció del robot, s'havia de comprovar també el funcionament dels motors; a través d'un programa proporcionat pels professors de la universitat, que s'introduïa en la placa connectada correctament als motors amb una petita peça mòbil als seus extrems, en lloc de les rodes. Llavors, per mitjà d'un programa de l'ordinador governat pel teclat del mateix, s'ordenava a ambdós dispositius un dels set moviments establerts: endavant, endarrere, parat, gir a la dreta, gir a l'esquerra, gir tancat a la dreta i gir tancat a l'esquerra.

Figura 10. Imatge del motor (Font d'Internet).

A continuació, s'havia de construir la carcassa del robot al voltant de la placa; utilitzant els tornavisos plans i en punta d'estrella, el martell i les alicates. L'esquema del procés de construcció que vam seguir és el següent:

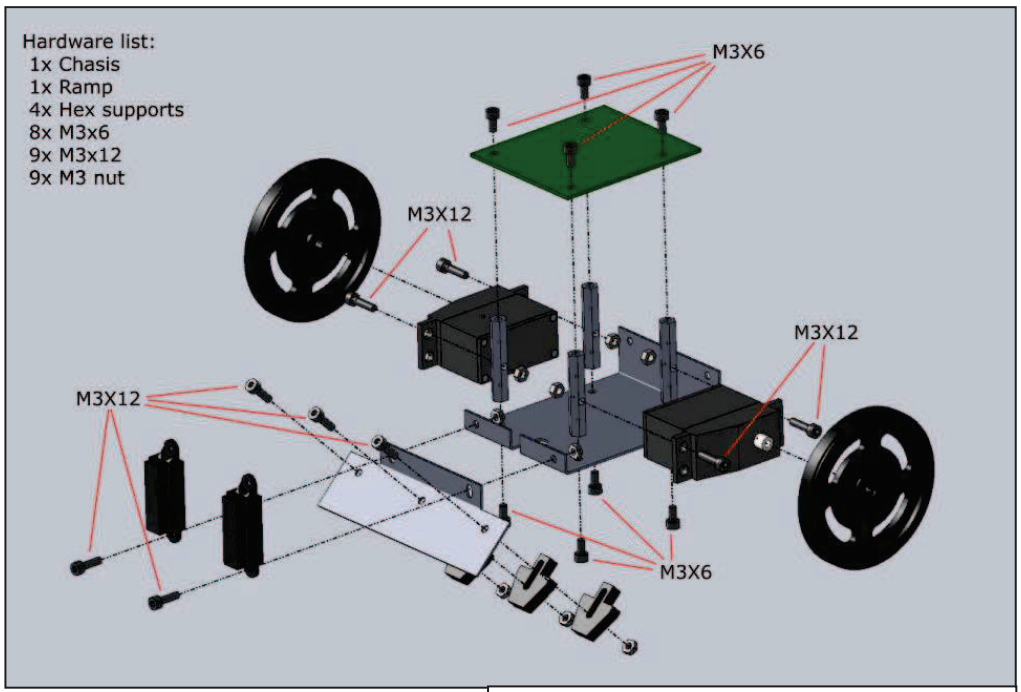


Figura 11. Esquema del muntatge del robot (Font proporcionada per l'UPC).

Un cop finalitzada la seva construcció, vam afegir quatre piles de 1,5V i una de 9V, connectades amb els cables vermells i negres de manera adequada.

Finalment, s'havia de començar la programació del robot, en funció del seus sensors. D'aquesta manera, havíem de decidir que havia de fer el robot en cas d'arribar al terra fosc i al clar i també en cas de detectar altres cossos a certa distància: avançar, anar endarrere, parar-se, girar a la dreta, girar a l'esquerra, girar a la dreta sobre si mateix o girar a l'esquerra sobre si mateix. Aquest programa també es troba en l'apartat 8.6 d'annexos.

Per acabar el curset, vam fer tots plegats una competició per veure quin era el robot amb la programació més adequada per convertir-se en el campió del torneig de sumo. Tot i que vam patir un petit problema amb els sensors de distància a l'hora de competir, vam poder resoldre'l i aconseguir una bona posició en la classificació final del campionat.

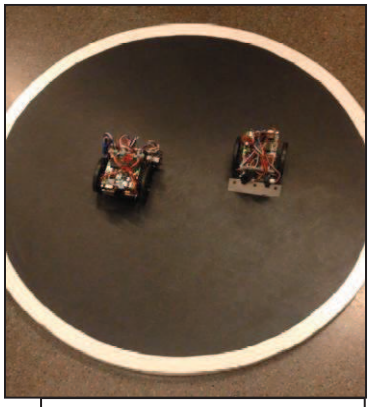
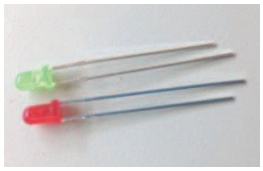


Figura 12. Robots competint (Font pròpia).

En l'USB que hi ha al final del treball, hi ha reflectit gran part d'aquest procés en forma de vídeo.

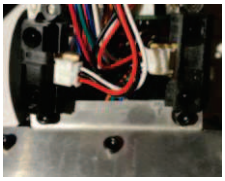
3. Procés de construcció

3.1. Materials i eines

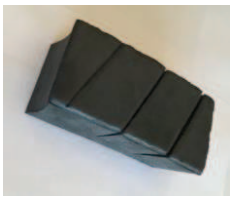
MATERIALS	NOMBRE D'UNITATS	IMATGE	FUNCIÓ
Condensador (0.1uF axial capacitor 50V Z5U)	10		Emmagatzema energia en el camp elèctric que s'estableix entre un parell de conductors, els quals estan carregats amb càrregues elèctriques oposades.
Condensador electrolític d'alumini (100uF radial aluminum electrolytic capacitor 16V)	2		Tipus de condensador elèctric, de 16 volts, que fa servir un líquid iònic conductor, amb més capacitat per unitat de volum que els altres.
Condensador electrolític d'alumini (33uF radial aluminum electrolytic capacitor 6.3V)	2		Tipus de condensador elèctric, de 6.3 volts, que fa servir un líquid iònic conductor, amb més capacitat per unitat de volum que els altres.
Díode de canvi ràpid (1N4148 Fast Switching Diode)	1		Component electrònic que només deixa circular el corrent elèctric en un únic sentit i el bloqueja en el sentit contrari.
Leds de color vermell i verd	2		És un component electrònic passiu; concretament, un díode que emet llum verda o vermella en funció d'alguna condició imposada.

40 pin 100 mil pitch dual-row header	1		Component format per 40 pins (terminals en forma de clavilla amb la funció de connectar components sense necessitat de soldar res) que aconsegueix transferir electricitat i informació.
100 mil pitch 5-pin header	1		Component format per 5 pins.
100 mil pitch 3-pin header	2		Component format per 3 pins.
2.0mm pitch 3-pin shrouded header	2		Component format per 3 pins.
DB9 Female PCB vertical mount	1		Connector amb la funció de connectar la placa amb l'ordinador.
6-pin 100 mil pitch dual-row header	1		Component format per 6 pins.
Resistències (1/8W 5% resistor)	14		Component electrònic que ofereix resistència al pas del corrent elèctric.
Sòcol (600 mil 40 pin DIP socket)	1		És un sistema electromecànic de suport i connexió elèctrica, instal·lat a la placa base, que s'usa per a fixar i connectar un microprocessador.
Interruptor	1		Dispositiu que permet desviar o interrompre el curs d'un corrent elèctric.

Polsador	1		Element que permet el pas o la interrupció del corrent elèctric mentre és accionat.
Clema (Terminal Block 3 positions 3.5mm pitch)	1		Tipus de connector elèctric amb tres connexions en que un cable s'empresona contra una peça metàl·lica mitjançant l'ús d'un cargol.
Microcontrolador (PIC16F877 DIP 40)	1		Circuit integrat programable, capaç d'executar les ordres gravades en la seva memòria.
250mA LDO Regulator	1		Regulador de tensió amb l'objectiu de protegir aparells elèctrics i electrònics delicats de variacions de diferència de potencial. Estan presents en les fonts d'alimentació de corrent continu regulades. La seva missió és la de proporcionar una diferència de potencial constant a la seva sortida.
Dual RS-232 Transmitter/ Receiver	1		Dispositiu que conté dos conductors i dos receptors, juntament amb un conjunt de pins de funcions determinades.
20.00MHz Resonator w/caps	1		Component electrònic que genera vibracions mecàniques ressonants en el dispositiu.

Pintura	3		Material que vam utilitzar per decorar la maqueta de color negre, vermell i blanc.
Fusta	12		Material que forma la maqueta.
Estructura del robot	3		Parts metàl·liques que donen forma al robot.
Rodes	2		Part que permet la mobilitat del robot.
Sensors de línia	3		Rastrejador que detecta els colors de la línia a seguir a partir d'un sensor d'infraroigs reflectiu.
Sensors de distància	2		Rastrejador que detecta els cossos a distància a partir d'un sensor d'ultrasons.
Servomotors	2		Dispositiu actuador que té la capacitat de situar-se en qualsevol posició dins del seu rang d'operació, i de mantenir-se estable en aquesta posició.
Cargols i femelles	34		Elements mecànics utilitzats per fer una unió desmuntable de peces diferents.

Placa	1		Targeta de circuits impresos.
Cables	Indefinits		Element conductor format per un conjunt variable de fils metàl·lics, recoberts per un material aïllant i amb la finalitat del transport de electricitat.
Bateria	1		Dispositiu de 9V que genera energia elèctrica.
Piles	4		Dispositiu de 1,5V que genera energia elèctrica.
Estany	Indefinit		Material que s'utilitza per fer soldadures.
Cinta adhesiva protectora	1		Material que emmascara àrees que no han de ser pintades.
Cola	1		Substància líquida que permet enganxar diversos materials.
Escuradents	Indefinits		Estri de fusta, allargat i prim.

Cinta adhesiva verda	1		Material que emmascara àrees i simula la gespa.
Poliestirè expandit	2		Material plàstic escumat.

EINES	IMATGE	FUNCIÓ
Tornavís		Eina que permet cargolar i descargolar caragols.
Multímetre		Instrument de mesura electrònic que mesura magnituds elèctriques (voltatges i intensitats).
Pinzell		Eina que serveix per pintar superfícies.
Serra de marqueteria		Eina que serveix per serrar.
Alicates		Eina multifunció utilitzada per fer de palanca o per fer girar cargols femella.
Ordinador		Màquina electrònica que rep i processa dades per a convertir-les en informació útil.

Cable COM-USB		Eina que serveix per transmetre informació entre un port USB i un COM.
Soldador		Eina que serveix per fer soldadures.
Llima		Eina que serveix per tallar.
Pelacables		Eina que serveix per separar la funda de plàstic del coure en els cables elèctrics.
Trepant		Màquina que s'utilitza per perforar materials diversos.
Pistola de silicona elèctrica		Eina que s'utilitza per escalfar la silicona, un material fortament adhesiu.

3.2. Construcció de la maqueta

Per demostrar les funcions del robot, vam decidir crear un petit circuit on quedessin exposades les seves facultats. Per donar-li una mica d'originalitat, també vam inventar un context on la maqueta simulava un tros de la carretera d'una ciutat amb un parc de bombers i una casa incendiada i on el robot representava una camió de bombers.

Així doncs, vam comprar una gran superfície de fusta contraxapada i la vam tallar en peces per formar el recorregut de la carretera. Les dimensions d'aquestes estan reflectides en els plànols, adjuntats al final del projecte. A més, vam dissenyar la casa en flames, el parc de bombers i la carcassa del robot.

En primer lloc, vam llimar les parts del circuit i vam dibuixar amb llapis la zona que corresponia a l'asfalt. Tot seguit i amb l'ajuda de la cinta adhesiva, vàrem pintar de negre l'àrea marcada anteriorment. Per últim, vam folrar la resta de la superfície, envernissada amb cola de contacte, amb una cinta adhesiva que simulava la gespa. D'aquesta manera, aconseguíem unir a la vegada totes les peces del circuit.

Per altra banda, no vàrem poder decorar la maqueta amb altres elements característics, tal i com teníem pensat; donat que el robot no pot detectar cap altre cos al seu voltant.

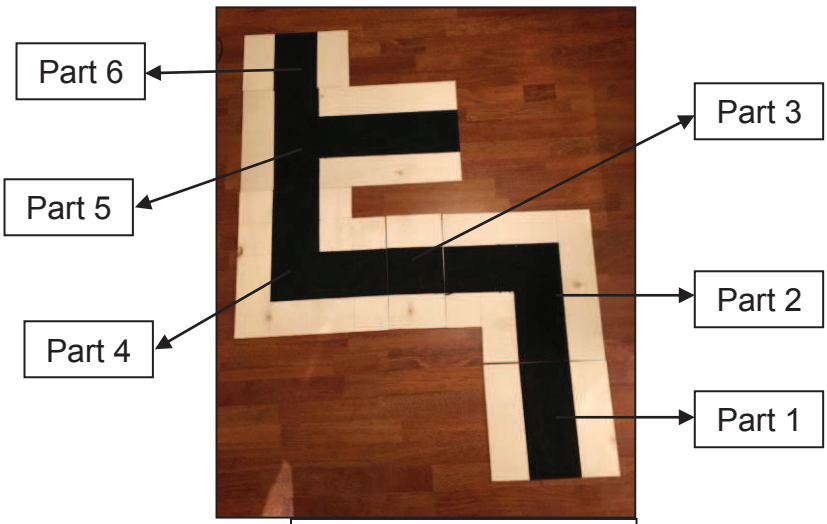


Figura 13. Imatge de la maqueta sense decorar (Font pròpia).

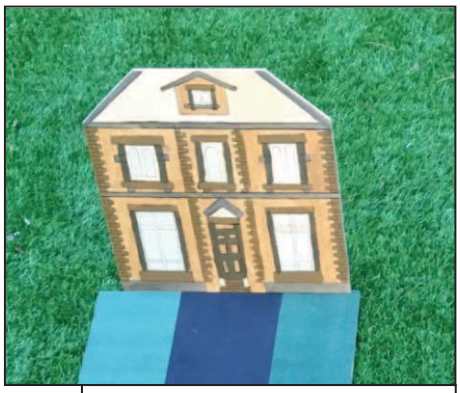


Figura 14. Imatge de la casa clàssica (Font pròpia).

En segon lloc, vam dissenyar una casa clàssica i un pàrquing de bombers sobre dos trossos de fusta rectangulars i les vam pintar de diferents colors. Per poder mantenir en posició vertical la mansió, vam construir un suport de fusta.

En tercer i últim lloc, vam construir la carcassa de la màquina amb quatre peces de contraxapat, pintades de vermell i unides amb cola transparent per a fusta. Després, vam unir una cinquena peça vermella per mitjà de dues frontisses; ja que era necessària la seva mobilitat per poder encendre l'interruptor de la placa. A més, vam enganxar un petit tros de poliestirè expandit en l'altre costat per evitar que la peça quedés desequilibrada.

Per fixar el robot amb la seva carcassa, vam enganxar a l'estructura del robot uns petits suports de fusta amb silicona. Després, vam subjectar a l'interior de la carcassa, també amb silicona, dues peces de poliestirè expandit modelades adequadament. Tot seguit, vam afegir, novament amb cola de contacte, dos suports de poliestirè expandit modelats en l'estructura davantera del robot per millorar la seva estabilitat.

Figures 15, 16 i 17. Imatges de la carcassa i dels seus suports (Font pròpia).

A continuació, vam foradar, amb una broca de 6 mm de diàmetre i una llima, dos rectangles en la part davantera de la carcassa, amb l'objectiu d'alliberar els sensors de distància.

Figura 18. Imatge de la carcassa amb els forats (Font pròpia).

Després, vam acabar el disseny de la carcassa dibuixant-hi, amb pintures de colors, alguns dels elements més representatius d'un cotxe de bombers (llums davanteres i posteriors, finestres, para-xocs, placa).

Per altra banda, la primera intenció d'afegir una sirena i unes llums al cotxe de bombers, per mitjà de la pràctica del cotxe fantàstic, no va ser factible; perquè el robot construït només permet emmagatzemar un únic programa. Tot i així, vam idear un nou element decoratiu senzill: la construcció de dos escales a través d'escuradents i cola blanca, subjectades en la part de dalt de la construcció i també en la part del darrere.

Finalment, vam trobar un recipient diminut i segur per simular la bomba d'aigua i la mànega característiques del cotxe de bombers. Per situar-la correctament, vam enganxar-la sobre un pedestal de poliestirè expandit en el sostre del cotxe de bombers.

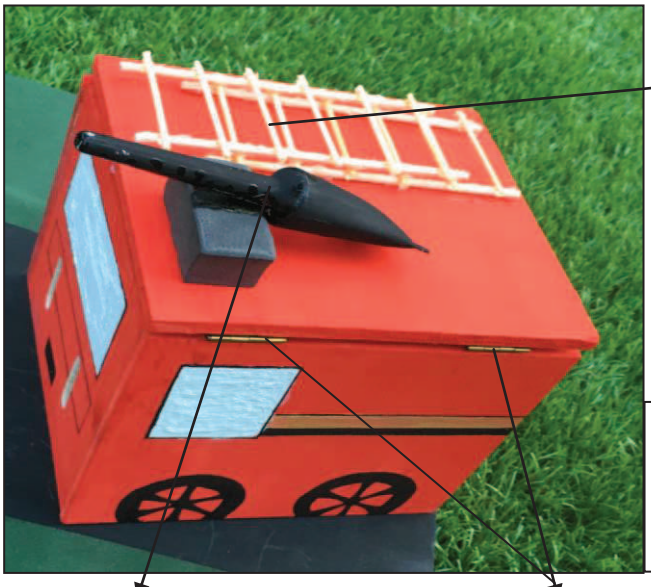


Figura 19. Imatge de la carcassa acabada (Font pròpia).

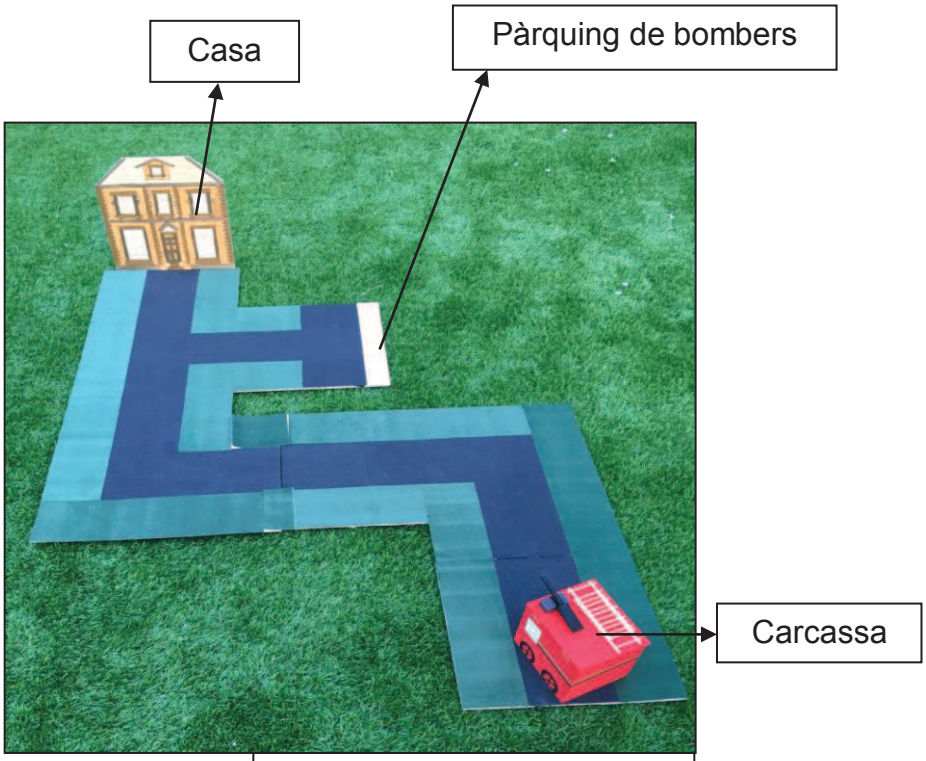


Figura 20. Imatge del circuit acabat (Font pròpia).

En l'USB adjuntat al final del projecte, es pot observar un vídeo on està reflectit tot aquest procés de construcció i també el de programació, explicat a continuació.

3.3. Segona programació del robot

Al canviar l'objectiu del projecte, era necessària la reprogramació del robot. La nova funció de la màquina era superar amb èxit el circuit de la maqueta: seguir una carretera amb diverses corbes, arribar fins a un edifici incendiat i quedar-se estacionat en un pàrquing.

Per crear el nou programa, va ser indispensable canviar la sentència de control de flux i totes les funcions que aquesta inclou. En aquest cas, vam poder reutilitzar les variables globals del programa del jugador de sumo, que són les que tenen el control dels motors. Però, el cos del programa principal (*void main*) va haver de ser totalment reprogramat.

El programa del cotxe de bombers comença cridant les llibreries i declarant les variables globals, que son les següents:

- 'aux': variable de tipus *int*.
- 'movimiento': variable de tipus *char* que serveix per saber quin tipus de moviment realitzarà el robot.
- 'liniaesquerra', 'liniacentral' i 'liniadreta': variables de tipus *int* en les quals es guarden els números que codifiquen els sensors de línia i que representen els diferents colors que detecten.
- 'ullsdreta' i 'ullsesquerra': variables de tipus *int* amb la funció de representar els números que codifiquen els sensors de llunyania segons la distancia en la qual es troben els objectes respecte d'aquests.

Seguidament, es troben les dues funcions dels sensors: la dels ulls i la de línia. Aquí es on es diu què han de fer aquests: agafar els valors segons el que detectin en cada moment i, a continuació, donar-los a la variable corresponent.

Més endavant, trobem la funció 'isr_timer1', que és l'encarregada de donar ordres segons el valor de la variable 'movimiento'. Està formada per un *switch*, on els valors dels diferents *case* són tots els tipus de moviments que pot fer el robot; i, dins de cada *case*, es troba el conjunt d'ordres que

executaran els motors en cada cas. Els valors de la variable 'movimiento' poden ser:

- 'a': el robot anirà endarrere.
- 'b': el robot pararà.
- 'c': el robot anirà endavant.
- 'd': el robot girarà amb les dos rodes a l'esquerra.
- 'e': el robot girarà amb les dos rodes a la dreta.
- 'f': el robot girarà amb una roda a l'esquerra.
- 'g': el robot girarà amb una roda a la dreta.

Per últim lloc, trobem la funció principal del programa, que utilitza la informació proporcionada pels sensors per saber quina orde donar al robot.

Així doncs, trobem inicialment un *while*, on la condició és sempre certa i, per tant, es declara un bucle infinit; seguit per un *if*, on en cas de que la variable 'liniacentral' detecti negre (si el seu valor és més gran que 30), el robot haurà de fer la sèrie de moviments establerts.

Després, està *else if*, on la condició és que els sensors de distància detectin un objecte a una certa proximitat (els valors de les variables 'ullsdreta' i 'ullsesquerra' han d'estar entre 40 i 70); i les sentències de dins estableixen que el robot faci una altra sèrie de moviments.

Finalment, trobem l'últim *else if*, amb la condició de que la variable 'liniacentral' detecti blanc (si el seu valor és més petit que 20). La seva sentència obliga al robot a quedar-se parat.

Aquest programa serveix únicament per a que el robot pugui seguir el recorregut marcat per la maqueta; ja que els moviments que es troben dins de cada condició tenen la durada exacta per a que aquest no es surti de la carretera i giri en el moment adequat.

En l'apartat 8.7 d'annexos, es troba el nou programa del cotxe de bombers.

3.4. Plànols

Els plànols del robot, fets amb AutoSketch9, estan dividits en dos exercicis, repartits a la vegada en diverses làmines:

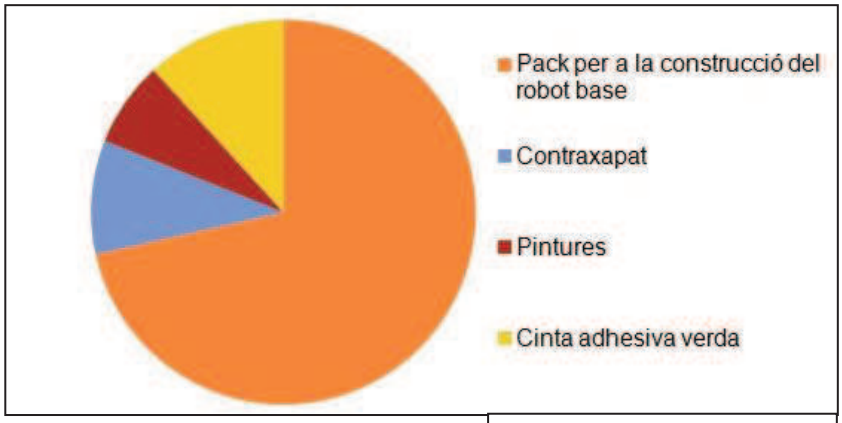
- Plànols del circuit (a escala ¼)
- Plànols de l'ambulància (a escala ½).

Aquests es troben impresos en DIN A3 al final del treball.

4. Pressupost

NOM	PREU PER UNITAT (€)	Nº D'UNITATS	PREU FINAL (€)
Pack per a la construcció del robot base	90	1	90
Contraxapat	12	1	12
Pintures	2,25	4	9
Cinta adhesiva verda	5,95	2,5	14,875
		PREU TOTAL (€)	125,875

Tot seguit, apareix un gràfic circular, on es reflecteixen les dades de la taula anterior en funció del seu preu:



Gràfic 1. Pressupost total.

Tal i com podem observar, el robot és el material més car i indispensable. Tot i així, els altres tres elements minoritaris (contraxapat, pintures i cinta adhesiva verda) també són necessaris per aconseguir crear el nostre projecte.

5. Temps dedicat

DATA	Nº D'HORES	FEINA REALITZADA
Del 29 de juny del 2015 al 3 de juliol del 2015	6 hores diàries	Curset de programació en C
6 de juliol del 2015	6 hores	Simulació d'un comptador d'una xifra
7 de juliol del 2015	6 hores	Simulació d'un comptador de dues xifres
8 de juliol del 2015	6 hores	Soldar la placa
9 de juliol del 2015	6 hores	Acabar de soldar la placa i comprovar-la
10 de juliol del 2015	6 hores	Pràctica del cotxe fantàstic
13 de juliol del 2015	6 hores	Display comptador d'una i dues xifres
14 de juliol del 2015	6 hores	Display comptador de segons d'una i dues xifres
15 de juliol del 2015	6 hores	Programació del robot

16 de juliol del 2015	10 hores	Muntatge del robot, tant pel matí com per la tarda
17 de juliol del 2015	6 hores	Acabar el robot
10 de setembre del 2015	3 hores	Comprovar el funcionament del robot, dissenyar el projecte i començar la memòria tècnica
4 d'octubre del 2015	3 hores	Continuar la memòria tècnica
16 d'octubre del 2015	4 hores	Comprar les fustes i tallar i llimar les de la carcassa del cotxe de bombers
2 de novembre del 2015	7 hores	Tallar i llimar les fustes del circuit i dibuixar el traçat
11 de novembre del 2015	4 hores	Continuar la memòria tècnica i fer els plànols
26 de novembre del 2015	3 hores	Continuar la memòria tècnica
8 de desembre del 2015	6 hores	Segona programació del robot i continuar la memòria tècnica

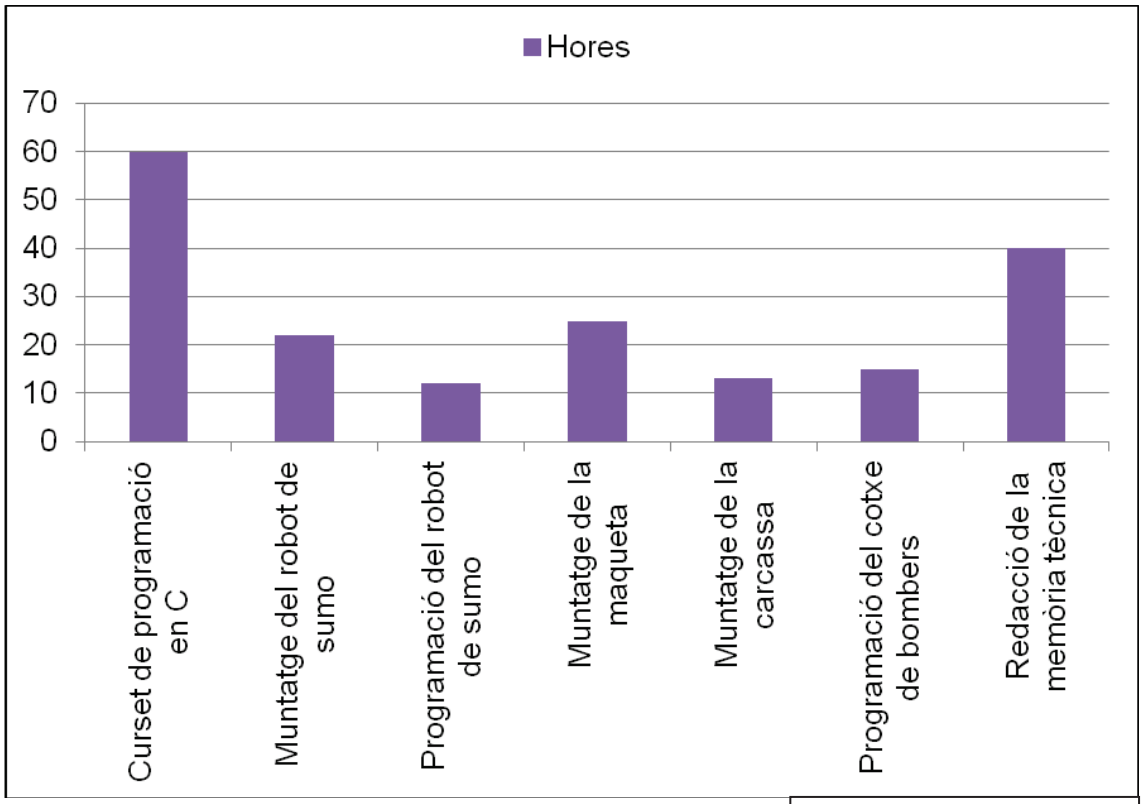
10 de desembre del 2015	1 hora	Primera capa de pintura a la carretera
11 de desembre del 2015	3 hora	Segona capa de pintura a la carretera i continuar memòria tècnica
12 de desembre del 2015	1 hora	Tercera capa de pintura a la carretera
13 de desembre del 2015	3 hores	Continuar la memòria tècnica
16 de desembre del 2015	2 hores	Continuar la memòria tècnica
20 de desembre del 2015	4 hores	Continuar la memòria tècnica
23 de desembre del 2015	4 hores	Programació del robot
27 de desembre del 2015	3 hores	Continuar la memòria tècnica
30 de desembre de 2015	6 hores	Continuar la memòria tècnica i la programació del robot
3 de gener de 2016	2 hores	Pintar les fustes de la carcassa

4 de gener de 2016	5 hores	Crear la casa en flames i el pàrquing de la maqueta
7 de gener de 2016	2 hores	Continuar la memòria tècnica
10 de gener de 2016	5 hores	Continuar la memòria tècnica i acabar l'estructura de la carcassa
13 de gener de 2016	3 hores	Fer el vídeo del projecte
17 de gener de 2016	6 hores	Continuar la memòria tècnica i acabar alguns detalls de la maqueta i la carcassa
18 de gener de 2016	6 hores	Acabar la maqueta i retocar la memòria tècnica
20 de gener de 2016	3 hores	Programació del robot i retocar la memòria tècnica

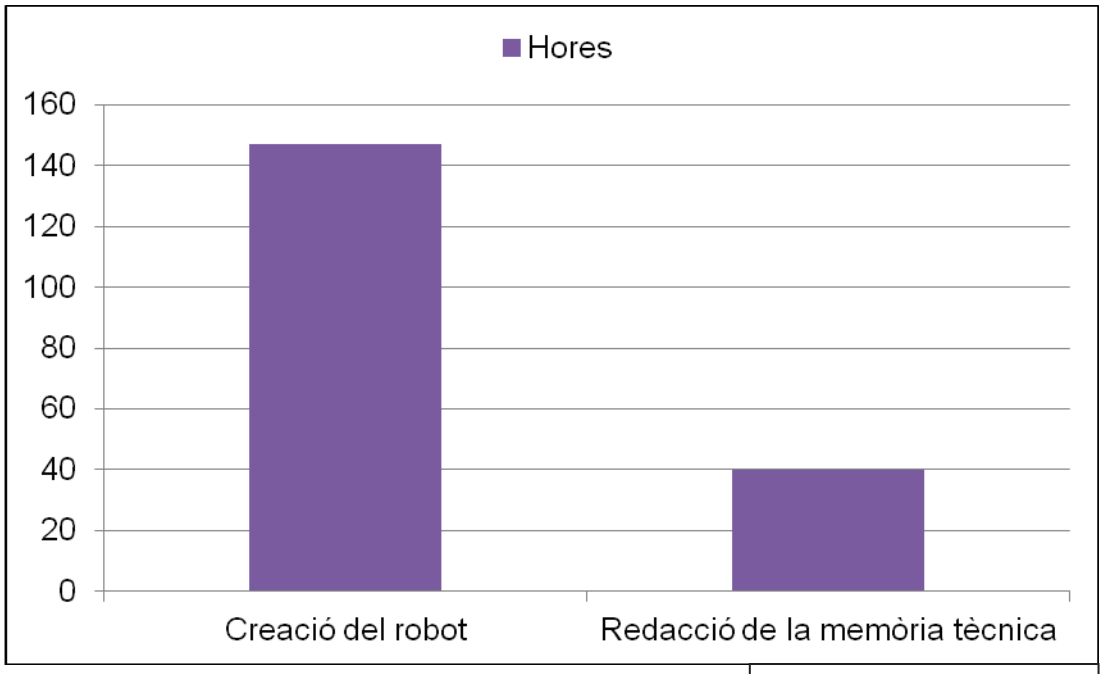
En conclusió, per fer aquest projecte s'ha hagut de realitzar un ampli conjunt de tasques i d'invertir nombroses hores de treball: el curset de programació en C (60 hores- incloent el desplaçament de Castelldefels a Barcelona i de Barcelona a Castelldefels en autobús), el muntatge del robot de sumo (22 hores), la programació del robot de sumo (12 hores), el muntatge de la maqueta (25 hores), el muntatge de la carcassa (13 hores), la programació del cotxe de bombers (15 hores) i la redacció de la memòria tècnica (40 hores).

En total, el temps invertit ha sigut aproximadament de 187 hores; de les quals el 79% ha estat dedicat a la creació del robot i tan sols el 21% a la redacció de la memòria tècnica.

A continuació, totes aquestes dades queden exposades en dos gràfics:



Gràfic 2. Temps dedicat.



Gràfic 3. Temps dedicat.

6. Conclusions i valoració crítica

Aquest treball de recerca ha estat molt laboriós, amb moltes hores emprades per aconseguir els nostres objectius; però destaca el fet de que ens ha servit per millorar diversos aspectes.

En primer lloc, ha estat un treball molt útil per conèixer detalladament el món de la robòtica; hem après molt sobre el funcionament intern del robot i dels diferents components que el formen, ja que hem estat nosaltres les encarregades de muntar-lo.

En segon lloc, hem après a programar, cosa que considerem que ha estat la part més complicada al llarg de tot el treball. Vàrem començar el curset de robòtica de la UPC sense cap idea de què era el llenguatge de programació en C; i, després de tres setmanes assistint-hi, vam acabar dominant el codi i vàrem ser capaces d'escriure, posteriorment, una programació per a que el nostre robot sigui capaç de superar el circuit que planteja la maqueta. A més, cal destacar el fet de que els programes de l'ordinador (*PIC C Compiler* i *PIC Downloader*) que vam utilitzar per a escriure la programació i per a poder introduir-la al robot van ser un gran repte per a nosaltres, ja que són bastant difícils de manejar.

En tercer lloc, hem aconseguit desenvolupar una part més creativa en un treball que en principi només es basa en un conjunt de coneixements tècnics i teòrics. Considerem que tant en la maqueta com en la carcassa ha estat una feina addicional el fet d'introduir elements artístics; però que ha valgut la pena, perquè fa que el projecte sigui molt més atractiu visualment.

Pel que fa a la recerca d'informació, no hem tingut cap problema; ja que al llarg del curset els nostres professors ens van donar la majoria de coneixements necessaris per a poder realitzar el treball. A més, ens van lliurar uns fitxers en PDF, d'on hem extret molta informació valuosa. Tota la resta d'informació la hem tret d'Internet, que no és gens difícil, ja que dades sobre l'electrònica i la robòtica hi ha un munt, encara que la gran majoria estan en anglès. Un cop

llegida tota la informació recaptada, vam haver de triar i sintetitzar les dades més adequades.

En relació amb les hipòtesis inicials plantejades ("Es pot aconseguir construir un robot seguidor de línies i capaç de guanyar una competició de sumo contra altres jugadors?" i "Es pot aconseguir construir un robot caracteritzat com si fos un cotxe de bombers i capaç de seguir un circuit que simuli el nucli urbà d'una ciutat fictícia?") considerem que sí que les hem aconseguit. Encara que hi ha alguns petits aspectes que hem hagut de modificar a causa de problemes tècnics, estem molt contentes amb els resultats finals i pensem que estan a l'altura de les nostres expectatives inicials.

Els inconvenients principals que ens van sorgir varen ser els següents:

- En primer lloc, mentre estàvem realitzant el curset en la UPC i també més tard programant per segona vegada el robot a casa, l'ordinador no era capaç de detectar la placa. Tot i així, ho vam poder solucionar canviant varies vegades d'ordinador i configurant els ports d'entrada i de sortida.

- En segon lloc, a l'hora de programar el robot de sumo, els sensors de distància no funcionaven adequadament. Per tal de solucionar el problema, vam haver de reajustar els intervals de proximitat en el programa.

- En tercer lloc, quan ja vam tenir acabada la placa, ens vam adonar que el botó de RESET estava trencat. Per tant, vam haver de dessoldar-lo i reemplaçar-lo per un de nou. Aquesta feina va resultar ser molt delicada, perquè cabia la possibilitat de cremar tota la placa si manteníem el soldador sobre la seva superfície massa temps.

- En quart lloc, la instal·lació dels programes adients per treballar amb l'ordinador de casa va resultar molt laboriosa i vam necessitar l'ajuda dels professors de la UPC. A més, el cable COM-USB per connectar la placa amb l'ordinador va ser difícil de trobar.

- En cinquè lloc, aquest tipus de motors es descalibren fàcilment i que tornin a funcionar adequadament comporta un gran treball i una notable pèrdua de

temps. Així mateix, en últim moment, quan el robot ja estava totalment acabat, el potenciòmetre (component electrònic que es troba a l'interior del motor) es va desviar i els motors van quedar descalibrats i, per tant, van deixar de funcionar. Així doncs, vam haver de desmuntar ambdós motors i posicionar correctament i amb enginy els seus potenciòmetres. Tot seguit, vam poder calibrar-los.

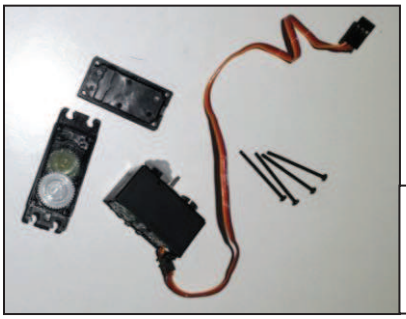


Figura 21. Imatge dels motors desmuntats (Font pròpia).

- Per últim lloc, no hem pogut aconseguir introduir la sirena i les llums d'alerta en el robot; ja que aquest té una capacitat d'emmagatzematge reduïda.

A més, en referència als objectius proposats en la introducció del treball, pensem que els hem superat tots amb èxit:

- Hem après a programar en llenguatge en C.
- Hem millorat força les nostres habilitats a l'hora de soldar components electrònics.
- Vàrem aconseguir fer un programa idoni per classificar-nos en una bona posició dins del torneig de sumo.
- Hem dissenyat de manera creativa la nostra maqueta.
- El nostre robot és capaç de superar tots els obstacles amb precisió.

En quant al treball en equip, hem aconseguit coordinar-nos i treballar d'una manera cooperativa, ajudant-nos mútuament en tot moment i esforçant-nos al màxim.

Finalment, només ens queda agrair a la UPC i a la Roser Blasco, el Marti Barri, la Meritxell Gómez i el Josep Izquierdo per exercir com a professors nostres al llarg de les tres setmanes del curset i per estar atents a les nostres preguntes i peticions durant els mesos posteriors; i també a la Pilar Ruiz, la nostra tutora del treball de recerca, que ens ha guiat i ajudat al llarg de tot el procés.

7. Fonts bibliogràfiques

MATERIAL SUBMINISTRAT PER L'UPC (adjuntat en el llapis USB)

1) Variables i constants
2) Operadors
3) Sentències de control de flux
4) Funcions
5) Llibreries

WEBGRAFIA

1) http://aess.upc.es/moodle/
2) https://www.google.es/imghp?hl=ca&gws_rd=ssl
3) https://en.wikipedia.org/wiki/Main_Page
4) http://www.areatecnologia.com/sistema-binario.htm
5) http://www.ite.educacion.es/formacion/materiales/47/cd/mod1b/1bb_4.htm

BIBLIOGRAFIA

1) PEÑA BASURTO, Marco A., CELA ESPÍN, José M. *Introducción a la programación en C.* Ediciones de la Universidad Politécnica de Cataluña. 2000.
2) BARRIENTOS, Antonio, PEÑIN, Luis Felipe, BALAGUER, Carlos, ARACIL, Rafael. *Fundamentos de robótica.* Segona edició. Espanya. S.A. McGraw-Hill. 2007. 9788448156367.

8. Annexos: Programació en C

8.1. Comptador d'una xifra

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

#define CERO 0b11101110
#define UNO 0b00101000
#define DOS 0b11001101
#define TRES 0b01101101
#define CUATRO 0b00101011
#define CINCO 0b01100111
#define SEIS 0b11100111
#define SIETE 0b00101100
#define OCHO 0b11101111
#define NUEVE 0b00101111

int num;
int
llista[10]={CERO,UNO,DOS,TRES,CUATRO,CINCO,SEIS,SIETE,OCHO,NUEVE};


void main (void)
{
  set_tris_d (0x00)

       while(1)
       {
         for (num=0;num<10; num++)
         {
           output_d(llista[num]);
         }
        num=0;
       }
}
```

8.2. Comptador de dues xifres

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)

#define CERO 0b00111111
#define UNO 0b00000110
#define DOS 0b01011011
#define TRES 0b01001111
#define CUATRO 0b01100110
#define CINCO 0b01101101
#define SEIS 0b01111101
#define SIETE 0b00000111
#define OCHO 0b01111111
#define NUEVE 0b01100111

int num,a=0,x;
int
llista[10]={CERO,UNO,DOS,TRES,CUATRO,CINCO,SEIS,SIETE,OCHO,NUEVE};

void main (void)
{
  set_tris_b(0x00);
  x=input(PIN_C0);
  output_bit(PIN_C1,x);
  if(x=1)
  {
   x=0;
  }else{
   x=1;
  }
      while(1)
      {
         for (num=0;num<10; num++)
         {
               while(a<10)
               {
                 output_low(PIN_C1);
                 output_high(PIN_C0);
                 output_b(llista[num]);
                 delay_ms(1);
                 output_low(PIN_C0);
                 output_high(PIN_C1);
                 output_b(llista[a]);
                 delay_ms(1);
                 a++;
               }
               a=0;
         }
    }
}
```

8.3. Cotxe fantàstic

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

void main (void)
{
  while (1)
  {
     output_high(PIN_D0);
     delay_ms(200);
     output_low(PIN_D0);
     output_high(PIN_D1);
     delay_ms(200);
     output_low(PIN_D1);
     output_high(PIN_D2);
     delay_ms(200);
     output_low(PIN_D2);
     output_high(PIN_D3);
     delay_ms(200);
     output_low(PIN_D3);
     output_high(PIN_D4);
     delay_ms(200);
     output_low(PIN_D4);
     output_high(PIN_D5);
     delay_ms(200);
     output_low(PIN_D5);
     output_high(PIN_D6);
     delay_ms(200);
     output_low(PIN_D6);
     output_high(PIN_D7);
     delay_ms(200);
     output_low(PIN_D7);
  }
}
```

8.4. Comptador de segons d'una xifra

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

#define CERO 0b01110111
#define UNO 0b00010100
#define DOS 0b10110011
#define TRES 0b10110110
#define CUATRO 0b11010100
#define CINCO 0b11100110
#define SEIS 0b11100111
#define SIETE 0b00110100
#define OCHO 0b11110111
#define NUEVE 0b11110100

int i;
int
llista[10]={CERO,UNO,DOS,TRES,CUATRO,CINCO,SEIS,SIETE,OCHO,NUEVE};

void main (void)
{
  set_tris_d (0x00);
    while(1){
    for (i=0; i<10; i++)
    {
        output_high(PIN_C0);
        output_low(PIN_C0);
        output_d(llista[segon]);
        delay_ms(400);

    }

}
```

8.5. Comptador de segons de dues xifres

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

#define CERO 0b01110111
#define UNO 0b00010100
#define DOS 0b10110011
#define TRES 0b10110110
#define CUATRO 0b11010100
#define CINCO 0b11100110
#define SEIS 0b11100111
#define SIETE 0b00110100
#define OCHO 0b11110111
#define NUEVE 0b11110100

int i;
int b;
int
llista[10]={CERO,UNO,DOS,TRES,CUATRO,CINCO,SEIS,SIETE,OCHO,NUEVE};

void main (void)
{
  set_tris_d (0x00);
     while(1){
     for (i=0; i<10; i++)
     {
        for (b=0; b<10; b++)
           {
           output_high(PIN_C0);
           output_low(PIN_C1);
           output_d(llista[segon]);
           delay_ms(400);
           output_high(PIN_C1);
           output_low(PIN_C0);
           output_d(llista[segon]);
           delay_ms(400);

           }
        }
     }

}
```

8.6. Programació del robot de sumo

```
#include <16f877.h>
#include <stdio.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP,PUT,BROWNOUT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

int aux=0;
char movimiento =' ';
int liniaesquerra, liniadreta, liniacentral;
int ullsesquerra, ullsdreta;

void linia(){

  set_adc_channel(5);
  delay_ms(1);
  liniadreta = read_adc();

  set_adc_channel(6);
  delay_ms(1);
  liniacentral = read_adc();

  set_adc_channel(7);
  delay_ms(1);
  liniaesquerra = read_adc();

  delay_ms(100);
  //printf("Linia:%u - %u - %u\n\r",liniadreta, liniacentral, liniaesquerra);
}
void ulls(){

  set_adc_channel(2);
  delay_ms(1);
  ullsdreta = read_adc();

  set_adc_channel(3);
  delay_ms(1);
  ullsesquerra = read_adc();
  delay_ms(100);
  //printf("Ulls:%u - %u\n\r",ullsdreta, ullsesquerra);

}

#INT_TIMER1
void isr_timer1(){
     switch (movimiento){

        case 'a':  //endarrere
             switch(aux){
                 case 0:
                      output_high(PIN_B1);
```

```
                output_high(PIN_B2);
                set_timer1(65224);
                aux = 1;
              break;

              case 1:
                output_high(PIN_B1);
                output_low(PIN_B2);
                set_timer1(64286);
                aux = 2;
              break;

              case 2:
                output_low(PIN_B1);
                output_low(PIN_B2);
                set_timer1(54598);
                aux = 0;
              break;
        }
    break;

    case 'b':  //parat
        switch(aux){
            case 0:
              output_high(PIN_B1);
              output_high(PIN_B2);
              set_timer1(64599);
              aux = 1;
            break;
            case 1:
              output_low(PIN_B1);
              output_low(PIN_B2);
              set_timer1(53974);
              aux = 0;
            break;
        }
    break;

    case 'c': //endavant
        switch(aux){
            case 0:
              output_high(PIN_B1);
              output_high(PIN_B2);
              set_timer1(65224);
              aux = 1;
            break;

            case 1:
              output_low(PIN_B1);
              output_high(PIN_B2);
              set_timer1(64286);
              aux = 2;
```

```
            break;

            case 2:
              output_low(PIN_B1);
              output_low(PIN_B2);
              set_timer1(54598);
              aux = 0;
            break;
        }
    break;

    case 'd':  //esquerra
        switch(aux){
            case 0:
                output_high(PIN_B1);
                output_high(PIN_B2);
                set_timer1(65224);
                aux = 1;
              break;
              case 1:
                output_low(PIN_B1);
                output_low(PIN_B2);
                set_timer1(53349);
                aux = 0;
              break;
        }
    break;

    case 'e':  //dreta
        switch(aux){
            case 0:
                output_high(PIN_B1);
                output_high(PIN_B2);
                set_timer1(63973);
                aux = 1;
              break;
              case 1:
                output_low(PIN_B1);
                output_low(PIN_B2);
                set_timer1(54598);
                aux = 0;
              break;
        }
    break;

    case 'f': //esquerra tancat
       switch(aux){
            case 0:
              output_low(PIN_B1);
              output_high(PIN_B2);
              set_timer1(65224);
              aux = 1;
```

```
                break;
                case 1:
                  output_low(PIN_B1);
                  output_low(PIN_B2);
                  set_timer1(53349);
                  aux = 0;
                break;
            }
        break;

        case 'g': //dreta tancat
            switch(aux){
                case 0:
                  output_high(PIN_B1);
                  output_low(PIN_B2);
                  set_timer1(63973);
                  aux = 1;
                break;
                case 1:
                  output_low(PIN_B1);
                  output_low(PIN_B2);
                  set_timer1(54598);
                  aux = 0;
                break;
            }
        break;
    }
}

void main(){
  set_tris_b(0x00);
  setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
  enable_interrupts(int_rda);
  enable_interrupts(INT_TIMER1);
  enable_interrupts(global);

  setup_adc(ADC_CLOCK_INTERNAL);
  setup_adc_ports(ALL_ANALOG);

  delay_ms(5000);

  while(1){
    ulls();
    linia();

        if (liniacentral<20)
        {
            movimiento='a';
            delay_ms(500);
            movimiento='d';
            delay_ms(500);
        }
```

```
        else if (liniadreta<20)
        {
            movimiento='a';
            delay_ms(700);
        }
        else if (liniaesquerra<20)
        {
            movimiento='a';
            delay_ms(500);
            movimiento='e';
            delay_ms(500);
        }
        else{
            movimiento='c';
        }

        if (ullsdreta > 40 && ullsdreta <70 && ullsesquerra > 40 && ullsesquerra <
70)
        {
          movimiento='c';

        }
        else if (ullsdreta > 70 && ullsdreta <100 && ullsesquerra > 70 &&
ullsesquerra < 100)
        { movimiento='g';
          delay_ms(500);
          movimiento='f';
          delay_ms(500);
          movimiento='c';
        }
        else if (ullsdreta&&ullsesquerra>100)
        {
          movimiento='c';
        }
    }
}
```

8.7. Programació del cotxe de bombers (programa definitiu del robot)

```
#include <16f877.h>
#include <stdio.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP,PUT,BROWNOUT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

int aux=0;
char movimiento =' ';
int liniaesquerra, liniadreta, liniacentral;
int ullsesquerra, ullsdreta;

void linia(){

  set_adc_channel(5);
  delay_ms(1);
  liniadreta = read_adc();

  set_adc_channel(6);
  delay_ms(1);
  liniacentral = read_adc();

  set_adc_channel(7);
  delay_ms(1);
  liniaesquerra = read_adc();

  delay_ms(100);
  //printf("Linia:%u - %u - %u\n\r",liniadreta, liniacentral, liniaesquerra);
}
void ulls(){

  set_adc_channel(2);
  delay_ms(1);
  ullsdreta = read_adc();

  set_adc_channel(3);
  delay_ms(1);
  ullsesquerra = read_adc();
  delay_ms(100);
  //printf("Ulls:%u - %u\n\r",ullsdreta, ullsesquerra);

}

#INT_TIMER1
void isr_timer1(){
     switch (movimiento){

        case 'a':  //endarrere
             switch(aux){
                 case 0:
                      output_high(PIN_B1);
```

```
                output_high(PIN_B2);
                set_timer1(65224);
                aux = 1;
              break;

              case 1:
                output_high(PIN_B1);
                output_low(PIN_B2);
                set_timer1(64286);
                aux = 2;
              break;

              case 2:
                output_low(PIN_B1);
                output_low(PIN_B2);
                set_timer1(54598);
                aux = 0;
              break;
        }
    break;

    case 'b':  //parat
        switch(aux){
            case 0:
              output_high(PIN_B1);
              output_high(PIN_B2);
              set_timer1(64599);
              aux = 1;
            break;
            case 1:
              output_low(PIN_B1);
              output_low(PIN_B2);
              set_timer1(53974);
              aux = 0;
            break;
        }
    break;

    case 'c': //endavant
        switch(aux){
            case 0:
              output_high(PIN_B1);
              output_high(PIN_B2);
              set_timer1(65224);
              aux = 1;
            break;

            case 1:
              output_low(PIN_B1);
              output_high(PIN_B2);
              set_timer1(64286);
              aux = 2;
```

```
            break;

            case 2:
              output_low(PIN_B1);
              output_low(PIN_B2);
              set_timer1(54598);
              aux = 0;
            break;
        }
    break;

    case 'd':  //esquerra
        switch(aux){
            case 0:
                output_high(PIN_B1);
                output_high(PIN_B2);
                set_timer1(65224);
                aux = 1;
              break;
              case 1:
                output_low(PIN_B1);
                output_low(PIN_B2);
                set_timer1(53349);
                aux = 0;
              break;
        }
    break;

    case 'e':  //dreta
        switch(aux){
            case 0:
                output_high(PIN_B1);
                output_high(PIN_B2);
                set_timer1(63973);
                aux = 1;
              break;
              case 1:
                output_low(PIN_B1);
                output_low(PIN_B2);
                set_timer1(54598);
                aux = 0;
              break;
        }
    break;

    case 'f': //esquerra tancat
        switch(aux){
            case 0:
              output_low(PIN_B1);
              output_high(PIN_B2);
              set_timer1(65224);
              aux = 1;
```

```
                break;
                case 1:
                  output_low(PIN_B1);
                  output_low(PIN_B2);
                  set_timer1(53349);
                  aux = 0;
                break;
            }
        break;

        case 'g': //dreta tancat
            switch(aux){
                case 0:
                  output_high(PIN_B1);
                  output_low(PIN_B2);
                  set_timer1(63973);
                  aux = 1;
                break;
                case 1:
                  output_low(PIN_B1);
                  output_low(PIN_B2);
                  set_timer1(54598);
                  aux = 0;
                break;
            }
        break;

    }

}


void main(){
  set_tris_b(0x00);
  setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
  enable_interrupts(int_rda);
  enable_interrupts(INT_TIMER1);
  enable_interrupts(global);

  setup_adc(ADC_CLOCK_INTERNAL);
  setup_adc_ports(ALL_ANALOG);

  delay_ms(5000);

  while(1){

    ulls();
    linia();

        if (liniacentral>30){

            movimiento='c';
```

```
            delay_ms(2000);
            movimiento='d';
            delay_ms(317);
            movimiento='c';
            delay_ms(2650);
            movimiento='e';
            delay_ms(283);
            movimiento='c';
            delay_ms(2750);
        }

      else if (ulls dreta > 40 && ullsdreta <70 && ullsesquerra > 40 &&
      ullsesquerra < 70){

            movimiento='b';
            delay_ms(500);
            movimiento='a';
            delay_ms(1050);
            movimiento='e';
            delay_ms(250);
            movimiento='c';
            delay_ms(2000);
        }

       else if (liniacentral<20){

            movimiento='b';
        }
}
```