NO TOQUIS L'INTERRUPTOR

Disseny, programació i construcció d'una caixa inútil

Marina Anento i Ferran Saigí

Segon de Batxillerat

Treball de recerca

INS Les Marines

Curs 2017/2018

Tutoritzat per Pilar Ruiz

“*It's supposed to be automatic, but actually you have to push this button*”

“Hauria de ser automàtic, però de fet has de pressionar aquest botó”

—John Brunner

Agraïments

Al Christian Soler, el nostre professor de robòtica, pel temps que ha dedicat a ajudar-nos a resoldre els problemes que ens han anat sorgint.

A la Roser Blasco, entre altres membres de l'AESS, per tots els coneixements que hem adquirit i el material que se'ns ha proporcionat.

A la Pilar Ruiz, la nostra tutora, per ajudar-nos a millorar aquest treball de recerca.

ÍNDEX

1. Introducció

En el nostre treball de recerca explicarem els passos que hem seguit per construir una *caixa inútil* amb Arduino a partir dels coneixements adquirits en un curs que vam fer a l'estiu a la Universitat Politècnica de Catalunya.
El nostre projecte consisteix en que després de l'accionament d'un commutador bipolar situat a la tapa d'una caixa de fusta, surt un dit mecànic que acciona el mateix interruptor en sentit contrari.

El plantejament inicial d'aquest treball era realitzar un curs de tres setmanes a la Universitat Politècnica de Catalunya i aprendre a programar en codi C sense experiència prèvia, a més de saber aplicar-lo al camp de la robòtica.
La nostra hipòtesi inicial era:
Amb hores de treball es pot aconseguir crear un robot lluitador de sumo que es pugi modificar i transformar en un resseguidor de línia
Durant les dues setmanes del curset de robòtica vam demostrar aquesta hipòtesi, i vam encaminar el nostre treball a millorar aquest robot complicant el programa. La professora del curs, la Roser Blasco, ens va proposar construir una *caixa inútil* i la idea ens va encantar.

Per tant, la nostra hipòtesi va canviar a la següent:
Es pot dissenyar, construir i programar una *màquina inútil* utilitzant el llenguatge d'Arduino a partir dels coneixements de C previs i aconseguir que funcioni
L'objectiu principal era construir el nostre propi robot, dissenyant-lo i programant-lo nosaltres, aprenent de forma autodidacta el llenguatge d'Arduino.

Des del primer dia vam decidir anotar-ho tot: els coneixements adquirits, tots els detalls del projecte per mes mínims que fossin, les propostes, els canvis realitzats respecte la proposta original, etc.
El nostre treball es divideix en dos parts ben diferenciades, la memòria escrita i la part pràctica. Un cop vam encaminar el nostre projecte, vam decidir que primer faríem la part pràctica, és a dir, el robot, i un cop l'acabéssim començaríem a redactar la memòria escrita.

Pel que fa a la motivació principal, és l'interès que desperta en nosaltres el món de la tecnologia. El departament de Tecnologia oferia aquest tema pel treball de recerca i el vam escollir perquè ningú de nosaltres no havia fet mai res semblant i vam veure una gran oportunitat d'aprendre.

2. Curs d'introducció a la programació en C i Robòtica

Com ja hem mencionat anteriorment, per dur a terme un primer contacte amb el món de la robòtica vàrem anar a la Universitat Politècnica de Barcelona. Primer vam fer un curset cent per cent teòric, en el que vam adquirir coneixements bàsics sobre la programació en C, i després un altre majoritàriament pràctic en el que vam construir un robot sumo.

2.1 Introducció a la programació en C

Per controlar el moviment d'una màquina es necessita un llenguatge que tant la màquina com l'ésser humà puguin entendre, Atenent aquesta necessitat es va crear el codi C, que permet especificar tant les dades amb les que treballarà com les accions que realitzarà.
Aquest llenguatge és el que nosaltres vam aprendre en el curs d'una setmana a la UPC, ja que només són necessàries unes poques instruccions per a aprendre a utilitzar les funcions bàsiques.

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

Constants i variables

Per una banda existeixen les constants, que tenen un valor fix i no poden ser modificades. Hi ha dos tipus de constants:

- **Constants literals:** el valor de les quals apareix directament en el codi cada vegada que és necessari per a una operació.
- **Constants simbòliques:** les quals són representades mitjançant un nom (símbol) en el programa. Per poder utilitzar el seu valor s'ha de fer servir el seu nom simbòlic, de la mateixa manera que ho faríem amb una variable.

L'estructura de qualsevol constant és la següent:

```
#define nom_constant valor_constant
```

Les variables, en canvi, són objectes on es guarden valors, els quals poden ser consultats i modificats durant l'execució del programa.

Per poder utilitzar una variable en un programa, la variable s'ha de declarar al principi del programa. Les sentencies de declaració de variables indiquen al compilador que ha de reservar cert espai de la memòria del programa per tal d'emmagatzemar una dada. Hi ha tres tipus de variables depenent de l'espai que es necessita:

- Tipus **int**: per declarar valor enters.
- Tipus **float**: per declarar valors reals.
- Tipus **char**: per declarar caràcters.

Operadors

Un operador permet processar i modificar el valor d'una variable.

Hi ha diferents tipus d'operadors:

- **Operadors d'assignació:** ens permet assignar a la variable un valor inicial.

  Exemple: x=1.
- **Operadors aritmètics:** ens permeten operar amb les constants i les variables.

  Exemple: x++ equival a sumar una unitat a *x*.
- **Operadors racionals:** s'utilitzen per crear condicions en sentències condicionals o iteratives.

  Exemple: x>=3 representa que *x* ha de ser major o igual que 3.
- **Operadors lògics:** s'utilitzen per elaborar condicions complexes.

  Exemple: x>=3 && x!=6 equival a que *x* ha de ser més gran o major a 3 i *x* no pot ser igual a 6.

Prioritat entre operadors

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

SENTÈNCIES DE CONTROL DE FLUX

Les estructures de control de flux són els elements que permeten que un programa realitzi unes accions o unes altres en funció de les condicions que s'estableixin.

Les sentències condicionals permeten executar un codi o un altre segons una determinada condició.

- La sentència *if* és la sentència condicional més utilitzada on s'executa un conjunt de sentències només si la condició es compleix.
- La sentència *switch* és una forma de sentència condicional, en la que s'executa un codi o un altre segons el valor que pren una expressió.

Les sentències iteratives executen un codi mentre una determinada condició es compleixi.

- La sentència *while* és la sentència iterativa més utilitzada ja que executa un codi un número de vegades indefinides mentre la condició es compleix (també anomenades "bucles").
- La sentència *do* és molt similar a la sentència *while*, però amb una diferència important: la condició de permanència s'avalua al final del bucle, no al principi; per tant, sempre s'executen les sentències del bucle almenys un cop.
- La sentència *for* és un bucle que, en C presenta molta més flexibilitat que els bucles *for* d'altres llenguatges de programació.

Les sentències d'entrada i sortida serveixen per mostrar dades per pantalla i introduir-ne per teclat.

- La sentència *printf* ens permet mostrar informació per pantalla.
- La sentència *scanf* ens permet introduir dades per teclat.

Llibreries i funcions

Les llibreries són una recopilació de funcions, arxius i dades que poden ser utilitzats en qualsevol programa en C incloent la llibreria de la següent manera:

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

D'altra banda, les funcions ens permeten modular el programa i fan que el seu desenvolupament sigui mes fàcil de portar a terme. Cadascuna de les funcions du a terme una senzilla operació amb variables (tant locals com globals) i constants. La funció s'ha de declarar al principi del programa, fora del codi principal.

Per crear una funció s'ha d'utilitzar aquesta estructura:

```
tipus nom_funció (variable)
        {
                sentències;
        }
```

Després, per poder utilitzar-la s'ha de "cridar la funció" en el moment que la necessitem, escrivint el següent en el programa:

```
nom_funció (paràmetres);
```

Estructura

Tot programa ha de seguir una estructura, independentment del llenguatge informàtic. En aquest cas, els programes escrits en C han de seguir la següent:

1. Declaració de llibreries.
2. Definició de constants.
3. Capçaleres i estructures de les funcions que utilitzarem.
4. Definició de variables globals.
5. Programa principal, introduït per l'estructura *void main*.

També es poden afegir notes o comentaris enmig del programa. Per fer-ho, simplement s'ha d'introduir a l'inici de qualsevol comentari els signes //.

2.2 Curs d'introducció a la robòtica

Després del curs de C, necessitàvem connectar el microxip a una placa per a començar a fer les primeres pràctiques.

El primer que vàrem fer per construir el robot va ser comprovar que no ens faltava cap component. El funcionament de la placa no varia depenent de l'ordre de soldadura, però per comoditat vàrem seguir la llista que ens va proporcionar l'AESS[1]:

1- Resistències
2- Díode
3- Condensadors ceràmics
4- Polsador de reset
5- Sòcols per a circuits integrats
6- LEDs
7- Tires de pins
8- Regleta de connexió amb cargol
9- Condensadors electrolítics
10- Port DB9
11. Clock
12. Interruptor
13. Connectors blancs (cal fer una petita modificació al del centre per a que encaixin)

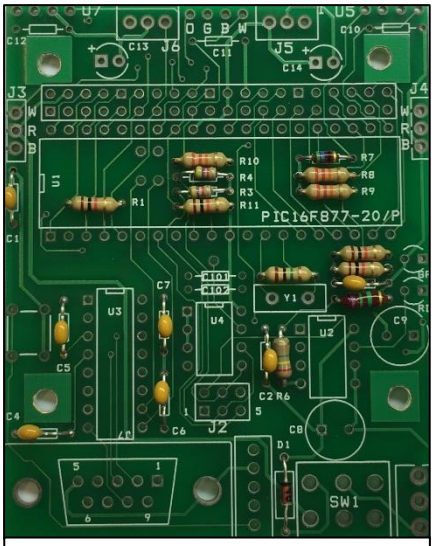


Figura 1. Placa amb les resistències, els díodes i els condensadors soldats.

Un cop soldats tots els components, vàrem connectar la placa a la font d'alimentació a través de la regleta. Com que tot estava ben soldat, el LED verd es va engegar. Posteriorment vam començar a reduir el voltatge i el díode vermell es va encendre. A continuació era necessari comprovar que el microxip funcionés adequadament. Vàrem connectar la placa a l'ordinador a través del port DB9[2], i vam configurar l'*hyperterminal*[3].

[1] L'AESS és l'associació d'estudiants de la Universitat Politècnica de Catalunya que es dedica al món de la robòtica i les noves tecnologies.
[2] El port DB9 és el que rep un connector analògic de 9 clavilles.
[3] L'hyperterminal és un programa que permet fer connexions Telnet. En el nostre cas va servir per comunicar-nos amb el robot, llegir el que s'havia de mostrar per pantalla o introduir dades.

Vam haver de seleccionar:

-El port COM

-Bits per second: 9600 bauds[4]

-Data bits: 8

-Parity: None

-Stop bits: 1

-Flow control: None

Aleshores, si el PIC[5] funcionava, es mostrava repetidament el missatge "alive" a la pantalla.

Un cop la placa va ser funcional, vam haver de posar en pràctica els coneixements adquirits durant el curs de C.
Abans de començar a picar el programa, era indispensable col·locar a la part superior tot allò necessari per a que funcionés en el nostre robot.

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
```

Sempre és necessari introduir la llibreria del nostre microxip al programa, utilitzant *#include <16F877.h>*. D'altra banda, la directiva *#fuses* s'utilitza per definir el comportament del PIC. S'indiquen al xip quins bits de la paraula de configuració volem activar. *HS* (High Speed) és per a que funcioni a màxima velocitat; *NOWDT* és per inhabilitar el temporitzador del *watchdog*[6]; *noprotect* és per no encriptar el nostre codi; *NOLVP* (no low voltage programming) és per definir que no programem amb un voltatge baix. També cal incloure el tipus de *clock*[7] que s'utilitza, que en el nostre cas era un de 20MHz.

[4] El baud és una unitat que s'utilitza per mesurar la velocitat de les transmissions telegràfiques.
[5] El PIC és un microcontrolador fabricat per *Microchip Technology*.
[6] El *watchdog* és un mecanisme de seguretat que reinicia el sistema en cas de que aquest es bloquegi.
[7] El *clock* també es coneix com a oscil·lador de freqüència, i indica al microprocessador la velocitat a la que ha de treballar.

2.2.1 Proves realitzades

1. La primera prova que vàrem fer va ser **una simulació d'un comptador** que anava des del zero fins al deu, i per fer-ho primer vàrem definir totes les variables abans del *void main*[8]. Vam a assignar cada número una seqüència de ports, activats i desactivats per formar el número que definíem. A continuació vam utilitzar un *while* per anar connectant els nombres i desconnectant els anteriors, mitjançant els comandaments *output_d(XIFRA);* i *output_toggle(XIFRA)*; respectivament. Les xifres anaven del zero fins al nou, i després ho feia a l'inrevés.

El problema que va suposar aquest comptador era que en el simulador de *seven-segment*[9] que vàrem instal·lar es veia bé, però com que el xip ho processava a molta velocitat, és veia constantment un 0, ja que les xifres canviaven tan ràpidament que no es podien apreciar els nombres.
Per tant, vam crear un nou programa en el qual vam col·locar les variables dels números dins d'un vector[10] i vam utilitzar *interrupcions* d'un segon, d'aquesta forma funcionava a temps real.

2. Un cop assolit l'objectiu de crear un comptador a temps real, vam fer el mateix però **amb dues xifres,** és a dir, utilitzàvem dos *seven-segments.* El programa era molt semblant, però quan el segon *display* arribava fins al 9, tornava al 0 i es sumava una xifra més al primer *display*. Utilitzàvem la *multiplexació* per a reduir el nombre de pins utilitzats, és a dir, no estaven activats els dos *seven-segment* a la vegada. S'alternaven constantment, però com que el xip ho processava ràpid, no s'apreciava el canvi i semblava que tots dos funcionessin a l'hora.

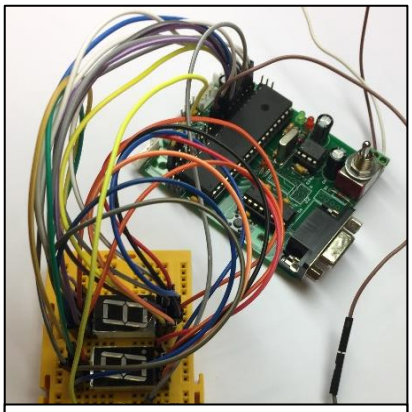


Figura 2. Connexions de la placa a la protoboard per realitzar el comptador de dues xifres.

[8] El *void main* és la part del programa que s'executa.
[9] El *seven-segment* és un dispositiu que serveix per a representar xifres en equips electrònics. Està compost de set segments que es poden encendre o apagar individualment.
[10] Un vector és una zona d'emmagatzematge contigua que conté una sèrie d'elements del mateix tipus.

3. El següent repte va ser crear **un programa que encengués 8 LEDs** col·locats a la *protoboard*[11] de manera que s'encenguessin quan s'apagava el LED del costat. Aquesta pràctica s'anomenava "el cotxe fantàstic". Cada díode anava associat a un pin. Vam crear una variable anomenada "x" que en cada sentència augmentava en una xifra. Aleshores, entrava en la següent condició, on imposàvem que s'apagués el pin anterior i que s'engegués el següent mitjançant els comandaments *output_toggle(PIN)* i *output_bit(PIN,1)*. A la última sentència donàvem a la x el valor de 0, i com que totes les condicions estaven dins d'un *while*, tot aquest procés es repetia infinites vegades.

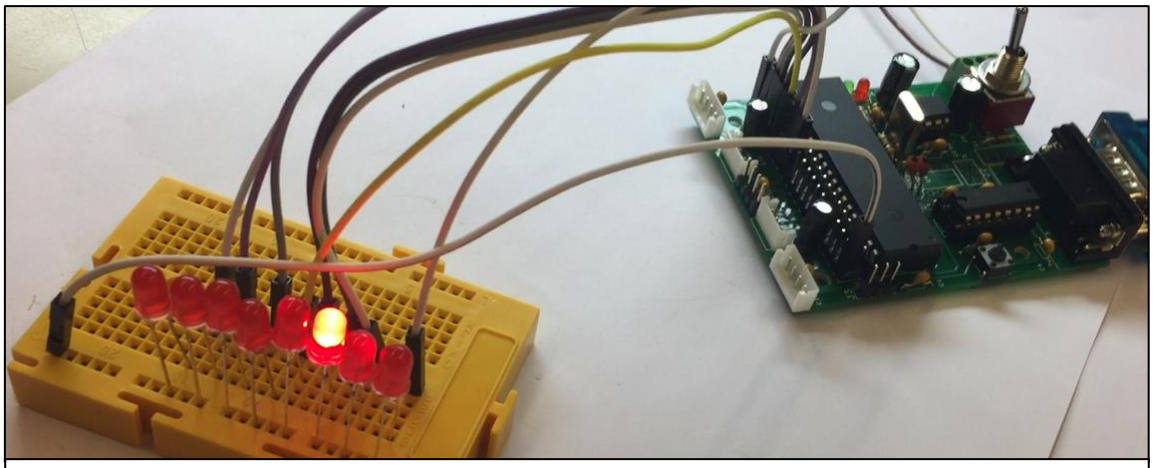


Figura 3. Connexions de la placa a la protoboard per realitzar el tercer repte.

4. Després vam rebre una classe teòrica sobre els motors més utilitzats en robòtica, i vàrem fer un **programa per aprendre a moure dos servomotors**[12], els que serien les nostres dues rodes. Al principi vam fer servir un *while*, però per economitzar i crear un programa més senzill vam utilitzar un *switch.* Depenent de la lletra inserida a l'*hyperterminal* entraven en un cas o en un altre.

[11] També coneguda com a placa de proves. Placa amb orificis que es troben connectats interna i elèctricament entre ells.
[12] Un servomotor és similar a un motor de corrent continu, però té la capacitat de situar-se en qualsevol posició dins del seu rang d'operació, i mantenir-se estable en aquesta.

2.2.2 Construint el robot lluitador de sumo

1. Per a les rodes, vam haver de **trucar els servomotors** per a que tinguessin rotació contínua, és a dir, que giressin més de 180º.
Primer vam extreure els quatre cargols per a poder desmuntar les tapes, i tot seguit el grup d'engranatges reductor. A l'engranatge de sortida vàrem trepar un forat amb una broca de 2mm a l'eix de rotació de la peça, per així poder calibrar els motors amb un tornavís posteriorment.

Figura 4. Procés de modificació del potenciòmetre del servomotor.

També vàrem eliminar la pestanya de fi de cursa, que bloquejava el motor quan aquest arribava als 180º, per fer-ho vam utilitzar una serra de disc i una llima elèctrica cilíndrica.
A continuació vàrem reutilitzar la serra de disc per fer una petita ranura (2mm de profunditat com a màxim) al centre del potenciòmetre.

Per a dir que aquest procediment s'ha realitzat correctament, el tornavís s'hauria de poder introduir al forat que hem fet l'engranatge de rotació i permetre girar el potenciòmetre.

2. Després vam haver de començar a **experimentar amb els sensors**, ja que són els que permeten que el robot es pugui moure en més d'una direcció i reaccionar amb el seu entorn. L'AESS ens va proporcionar dos programes per començar a provar els sensors. Vam haver de relacionar la distància que mesuràvem amb els valors que apareixien a la pantalla, per introduir-los al robot posteriorment.

3. Era el moment de **muntar la carcassa del robot**. L'AESS ens va proporcionar el material necessari i el vam muntar seguint aquest esquema:

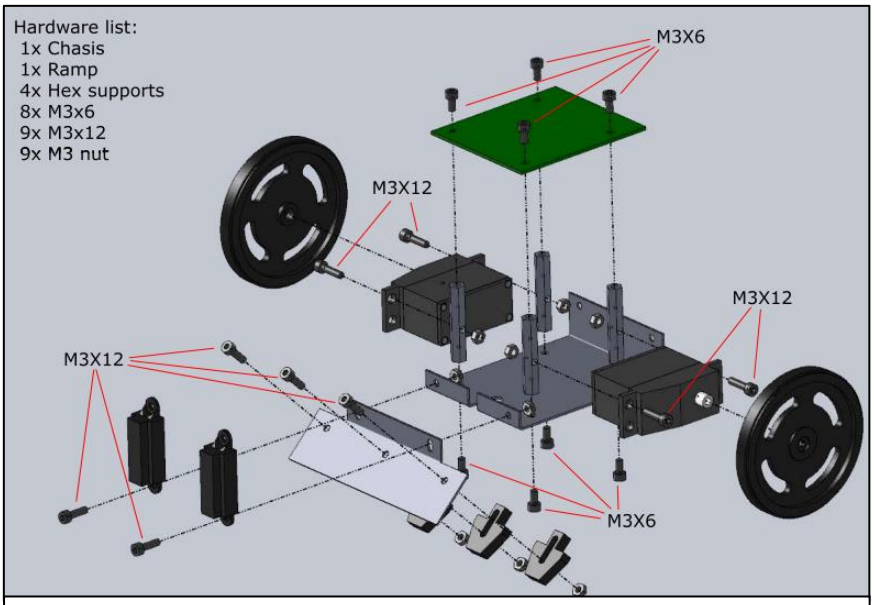
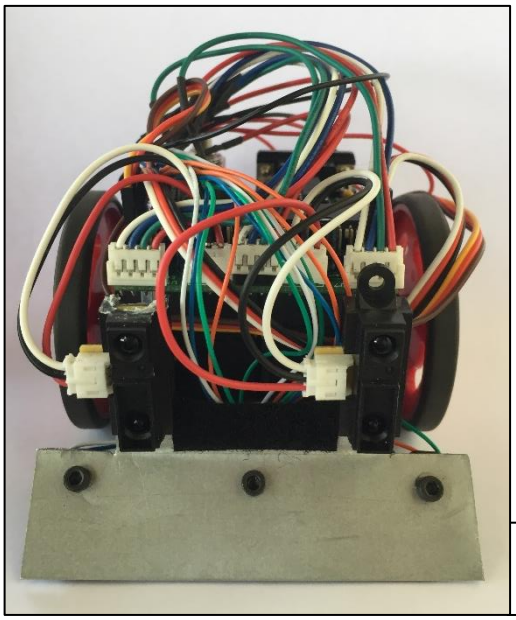


Figura 5. Instruccions pel muntatge del robot.



Després vam introduir les quatre piles d'1,5V al portapiles i el vam connectar a l'alimentació directa de la placa, i seguidament la pila de 9V per a l'alimentació dels servomotors.

Figura 6. Robot lluitador de sumo amb totes les connexions.

4. Per tal d'**amagar els circuits i millorar l'aspecte** del nostre sumo vàrem crear una coberta de pèl extraïble, enganxada al xassís del robot amb vetes adherents. Per a que els sensors funcionessin correctament vàrem retallar dos rectangles la tela. També vàrem fer un forat per a poder accedir a l'interruptor còmodament. Després vam cosir-li uns ulls per a donar al robot una millor aparença.

Figura 7. El nostre robot lluitador de sumo anomenat "La Bèstia".

Vam guanyar el concurs al millor sumo disfressat i vam rebre com a premi un kit d'Arduino, el qual utilitzàrem més tard pel segon robot.

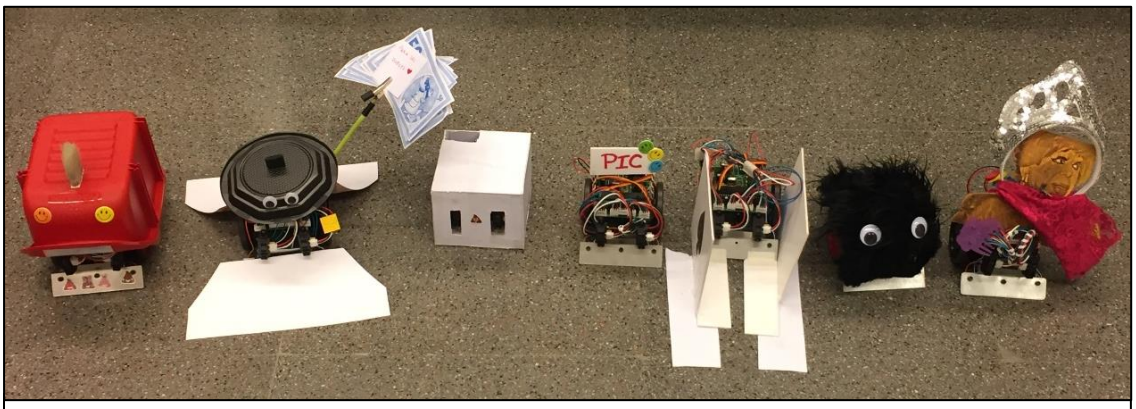


Figura 8. Robots de sumo disfressats.

3. Disseny, programació i construcció de la *caixa inútil*

3.1 Disseny

El primer que vàrem fer va ser començar a pensar de quines dimensions voldríem la caixa. Vam tenir en compte el xassís del robot sumo, ja que al principi el volíem introduir tal qual. Vam calcular que d'amplada hauríem de deixar com a mínim uns 10 mm d'espai respecte els motors, per evitar qualsevol problema. Per tant, d'amplada vam decidir que medís 150 mm.

Per estètica vam voler que la caixa fos rectangular, així que vam decidir que de llargada medís 270 mm. Pel que fa a l'alçada, vam mesurar el robot i li vam afegir 30 mm, determinant-la com 140 mm.
Com que el gruix del contraplacat és de 5 mm, la base havia de ser de 270x140 mm. A més, per introduir l'estructura del robot s'havia de retallar un quadrat de 60 mm de costat, amb un rectangle més a cada costat de 95x20 mm. (*Veg. plànols als annexos*).

Per fer la coberta vam decidir que utilitzaríem dues fustes enganxades per a que fos més gruixuda. Per a que encaixes la tapa, les mesures d'una de les fustes havien de ser iguals que les de la base. La fusta que aniria enganxada a la més petita, havia de ser 10 mm més llarga i ample que l'altre, perquè havia de cobrir les parets de 5 mm de gruix. Al centre, vam col·locar un cercle de 10 mm de diàmetre on aniria l'interruptor. A més, també vàrem decidir que a 160 mm del lateral faríem la ranura per a que la caixa s'obrís.

Vam fer una caixa amb unes mides força estandarditzades perquè el projecte no estava del tot definit i necessitàvem marge per fer les modificacions necessàries.

3.2 Programació

3.2.1 Introducció a l'Arduino

L'Arduino és una plataforma de desenvolupament oberta, dissenyada per a la creació de prototips i d'aplicacions Hardware. Va ser creat inicialment per a estudiants, ja que abans d'això las plaques que existien eren cares.

La creació d'Arduino com un sistema que no necessita llicències i amb el seu propi llenguatge senzill, va donar com a resultat un entorn de desenvolupament molt proper al públic, facilitant l'ús de l'electrònica en projectes de qualsevol tipus i constituint una de les comunitats de desenvolupament més grans que existeixen a Internet actualment.

La plataforma es basa en el llenguatge C i suporta totes les seves funcions bàsiques, a més d'algunes de C++.

Nosaltres hem utilitzat la placa Arduino UNO, basada en el microcontrolador ATmega328. Té 14 pins d'entrada digital (dels quals 6 poden ser utilitzats com a sortides de PWM[13]), 6 entrades analògiques, un port de connexió d'USB, un encapçalament d'ICSP[14] i un botó de reinici. Per compilar un programa a la placa senzillament s'ha de connectar-la a un ordinador amb un cable d'USB, i per alimentar-la es necessita un adaptador de DC o pila. Es pot programar la placa UNO sense haver-te de preocupar de cometre cap error, en el pitjor dels casos s'hauria de reemplaçar el xip i reiniciar la placa prement el botó de reset.

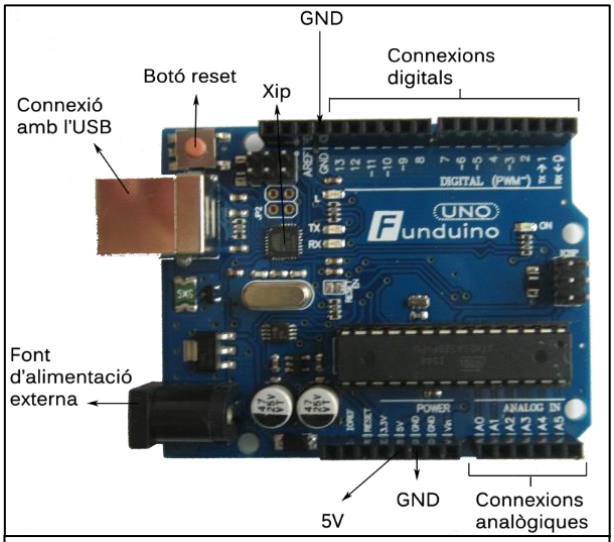


Figura 9. Indicacions de les parts de la placa.

[13] La modulació per amplada de polsos és una tècnica emprada per a codificar un missatge en una sèrie de polsos d'amplada variable. L'aplicació més usada és la dosificació de potència en fonts d'alimentació commutades.

[14] L'entrada ICSP (In Chip Serial Programmer) té accés a la memòria Flash. Es pot gravar directament un programa des del PC al microcontrolador sense utilitzar el port USB.

Objectes

Com hem explicat anteriorment, els objectes són diferents mecanismes que emmagatzemen un conjunt de valors necessaris en els diferents programes de l'ordinador. Aquests objectes poden ser constants o variables.

Constants

Les constants són expressions pre definides en el llenguatge Arduino. S'utilitzen per a fer els programes mes fàcils de llegir. Les constants més comunes són:

- *INPUT/OUTPUT*: representa l'entrada i la sortida de la senyal.
- *HIGH/LOW*: representen el nivells alt i baix de les senyals d'entrada i de sortida, on s'entén com a nivell alt 3 volts o més.
- *false* (fals): representa que el valor de la senyal és 0.
- *true* (vertader): qualsevol número diferent a zero és "cert", si és zero és "fals", segons l'àlgebra de Boole.

Tipus de dades

El tipus de variables depenen del tipus d'informació que es vulgui emmagatzemar i de la mida màxima d'aquesta. Existeixen diferents tipus de dades:

- *boolean*: ens indica simplement si la informació és certa o falsa.
- *byte*: emmagatzema una valor numèric de 8 bits i aquest pot estar entre el 0 i el 255.
- *char*: emmagatzema un número amb signe, entre -128 i 127. En alguns casos el compilador intentarà interpretar aquest tipus de dada com un caràcter.
- *unsigned char*: igual que el byte, el *unsigned char* codifica números entre el 0 i 255.
- *int* (16 bits): emmagatzema número amb signe entre el -32768 i el 32767.
- *unsigned int*: depenent del compilador el *unsigned int* pot ocupar 2 *bytes* (16 bits) i codificar números entre el 0 i el 65.535, o pot ocupar 4 *bytes* (32 bits) i codificar números entre el 0 i el 4.294.967.294
- *long*: codifica un valor entre el - 2.147.483.648 i el 2.147.483.647.
- *unsigned long*: emmagatzema valors sense signe entre el 0 i el 4.924.967.295. Aquest tipus de dada s'utilitza per emmagatzemar un

nombre en funció de las mil·lèsimes, la qual retorna el temps que el codi actual ha estat actuant en mil·lisegons.

- *float*: codifica valors que necessiten expressar-se en notació científica. Els valors han d'estar entre el -3,4028235 x 10^{38} i el 3,4028235 x 10^{38}.

Operadors

Un operador és un element del programa que s'aplica a una o diverses variables o constants en una expressió o instrucció. Un operador és un símbol que indica al compilador que s'ha de realitzar certes funcions matemàtiques. Depenent de quines funcions vulguis utilitzar els operadors es divideixen en diferents grups:

- Operadors aritmètics: Efectuen les funcions més bàsiques, la suma, la resta, la multiplicació i la divisió.
- Operadors compostos: aquestes funcions combinen una operació aritmètica amb una variable assignada. Aquestes expressions s'utilitzen bastant en els bucles.
- Operadors de comparació: s'encarreguen de comparar constants i variables amb unes altres per saber si una comparació és correcte.
- Operadors lògics: s'encarreguen de comparar dos expressions i retornar un VERDADER o FALS.

El llenguatge de programació Arduino va ser creat en certa manera a partir de del llenguatge de programació C++. És per això que els operadors i la prioritat d'aquests entre aquests llenguatges és molt semblant.

Estructures de control de flux

En programació, les estructures de control de permeten modificar el flux de execució de les instruccions del programa. Hi ha diferents tipus de estructures depenent del paper que exerceixen en el programa:

Les estructures seqüencials són aquelles en les que les instruccions són executades l'una darrere l'altre en un determinat ordre. Aquest ordre es pot alterar utilitzant estructures de Selecció o de Iteració, ja que permeten variar el control de flux del programa.

Les estructures de selecció permeten que es prenguin rutes d'acció alternatives depenent del resultat d'una condició. N'hi ha de diferents:

- El *if* és un estament que s'utilitza per provar si una determinada condició es compleix. Si la condició es compleix, el programa executarà una sèrie operacions que estan dins de les claus. Si la condició no es compleix el programa salta la estructura i no executa les operacions.
- *if else*: és una estructura que s'executa en resposta a la idea "si allò no es compleix fes això".
- *else*: pot anar precedit per una altra condició de manera que es poden establir varies estructures condicionals unes dins de les altres.
- *switch case*: igual que l'*if*, el aquesta estructura controla el flux del programa especificant en el programa quin codi ha d'executar en funció d'unes variables. Per tancar cada cas s'utilitza el *break*.
- *default*: estructura que serveix per executar un bloc d'operacions en cas de que no es compleixi cap de les condicions de les altres estructures condicionals.

Les estructures de repetició són funcions que s'utilitzen per repetir determinades operacions en cas de que les condicions es compleixin. Si la condició es compleix sempre, les operacions es repetiran indefinidament formant un bucle. Hi ha diferents estructures de repetició:

- *for*: aquesta estructura s'utilitza per repetir un bloc de sentències (operacions) tancades entre claus, un nombre determinat de vegades.
- *while*: és una estructura que crea un bucle d'execució continua mentre es compleixi la expressió col·locada entre parèntesis en la capçalera del bucle. Per sortir d'aquest bucle la variable de l'expressió haurà de canvia fent que la condició deixi de complir-se.
- *do while*: el bucle *do while* funciona de la mateixa manera que el bucle *while*, amb l'excepció de que la condició es provarà al final del bucle, de manera que el bucle sempre s'executarà com a mínim una vegada.
- *go to*: transfereix el flux del programa a un punt que està etiquetat.
- *break*: s'utilitza en les estructures *do*, *for* i *while* per sortir del bucle d'una forma diferent a la indicada.

- *continue*: s'utilitza en les estructures *do*, *for* i *while* per saltar la resta de les estructures que estan entre claus i continuï la següent execució del bucle comprovant l'expressió condicional.

Llibreries i funcions

Les llibreries són trossos de codi fets per altres persones que utilitzem en el nostres programes. Això, ens facilita molt la programació i fa que el nostre programa sigui més senzill de fer i d'entendre. Existeixen moltes llibreries desenvolupades per ajudar-te a connectar pràcticament qualsevol dispositiu al hardware d'Arduino.

Per utilitzar qualsevol llibreria abans s'ha de descarregar d'internet i instal·lar-la en cas de que encara no la tinguis. Seguidament, per poder utilitzar alguna funció de la llibreria en el programa, s'ha d'incloure al principi d'aquest escrivint exactament el mateix que escriuríem en un programa escrit en llenguatge C:

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

Per poder utilitzar qualsevol funció de la llibreria prèviament inclosa s'ha de "cridar-la" quan la necessitem escrivint el següent en el programa:

```
nom_funció (paràmetres);
```

Si es necessita crear una funció s'ha d'escriure aquesta estructura al principi del programa:

```
tipus nom_funció (variable)
        {
            sentències;
        }
```

Estructura

Per a que un el compilador detecti un programa, aquest ha de seguir una estructura igual que tots els altres programes escrits en el mateix llenguatge informàtic. En el cas d'Arduino, com que va ser creat a partir del llenguatge C++ l'estructura que segueixen aquests dos llenguatges per escriure un programa és la mateixa:

1. Declaració de llibreries.
2. Definició de constants i variables.
3. Capçaleres i estructures de les funcions que utilitzarem.
4. Definició de variables globals.
5. Programa principal

   La execució d'un programa es divideix en dos parts:

   - El *void setup*: constitueix la preparació del programa. Inclou la declaració i és la primera funció que s'executa en el programa, la inicialització de la comunicació en sèrie.
   - El *void loop*: constitueix la execució del programa. Inclou el codi a ser executat.

3.2.2 Creació del programa

Abans de començar a crear un programa, vam haver de redactar el que volíem que fes el nostre robot. Vam decidir que faríem cinc casos, i els vam enumerar. El programa ens hauria de donar nombres entre l'1 i el 5, i depenent de la xifra entraria en una condició o una altra. També vàrem enumerar els servomotors, així segons la distribució d'aquests i la condició podríem ordenar el que han de fer. L'esborrany inicial era el següent:

Un cop la placa rep alimentació:

Si s'ha activat l'interruptor:

Si la funció random dóna el nombre 1:

El servo1 gira 90º ràpid, i després -90º.

Si la funció random dóna el nombre 2:

El servo1 gira 90º lent (amb delays) i després -90º.

Si la funció random dóna el nombre 3:

El servo2 gira durant 2 segons, i un cop aquest para, el servo 1 gira 90º i

després -90º.

Si la funció random dóna 4:

El servo2 gira Xº(S'hauran de fer proves per a determinar els graus) i fa un delay d'un segon. Després el servo1 gira 90º i després -90º. Després el servo2 gira –Xº.

Si la funció random dóna 5:

El servo2 gira Xº i llegeix el valor del sensor. Si aquest valor és més gran que

70, els servos 3 i 4 giren durant dos segons.

El servo1 era el que havia de controlar el dit i els servos 3 i 4 servien per les rodes.

El servo2 ens permetria fer l'efecte visual de que la tapa s'obrís i es tanqués repetidament en pocs segons. Finalment vam decidir eliminar aquesta funció perquè no era estrictament necessària. També vam contemplar l'idea de posar sensors a la part interior de la tapa, de manera que quan el servo2 aixequés la

tapa, si es detectava una presencia, el robot aniria marxa enrere. No va ser possible utilitzar-los i va caler eliminar els sensors del programa a més del servo2 (*Veg. apartat 5*).

```
#include <Servo.h>
int vel3 = 0;
int vel4 = 0;
int maxnum = 110;
const int inputPin = 2;
const int inputPincalibrar = 5;
int calibrar = 0;
int value = 0;
int nombre;
Servo Servo1;
Servo Servo3;
Servo Servo4;
```

El que encapçala el nostre programa és tot allò que s'introdueix abans del *void setup*. Comencem introduïnt la llibreria *Servo.h*, el que permet crear les variables Servo, que el propi programa ja reconeix com a Servomotors.

Després tenim les variables *vel3* i *vel4*, que controlen els delays[15] de les funcions. A continuació hi ha les variables *inputPin* i *inputPincalibrar*, on s'associen els pins d'entrada dels interruptors i les variables *value* i *calibrar* guarden la informació proporcionada per aquests, respectivament. Per últim, a la variable *nombre* es guarda el número de l'1 al 10 proporcionat per la funció *random*.

Abans del *void setup* també s'han de declarar les funcions que s'utilitzaran al void loop.

```
void anar_endavant(int temps)
          {
            vel3 = 180;
            vel4 = 0;
            Servo3.write(vel3);
            Servo4.write(vel4);
            delay(temps);
          }
```

```
void anar_enrere(int temps2)
          {
            vel3 = 0;
            vel4 = 180;
            Servo3.write(vel3);
            Servo4.write(vel4);
            delay(temps2);
          }
```

Les funcions *anar_endavant(temps)* i *anar_enrere(temps2)* segueixen la mateixa estructura i tenen una variable existent únicament dins la pròpia funció, que permet canviar els *delays* quan es crida la funció al *void loop*. Amb la funció *Servo.write(graus)* pertanyent a la llibreria *Servo.h*, es pot indicar al motor quants graus es vol que giri. En aquest cas, els nostres servomotors giren en el sentit

[15] La funció delay() pausa el programa durant els mil·lisegons especificats dins del parèntesis.

que marquen les variables *vel3* i *vel 4* respectivament, i en sentits oposats per poder desplaçar el robot en el mateix sentit.

```
void parar()
          {
          vel3 = 90;
          vel4 = 90;
          Servo3.write(vel3);
          Servo4.write(vel4);
          }
```

Per a poder finalitzar el moviment, vam crear la funció *parar()*, que fa que els motors tornin a la posició inicial i romanguin quiets.

Tot seguit declararem les dues funcions més importants, les que controlaven el moviment del dit. Les funcions *dit_endavant(vel)* i *dit_enrere(vel2)* també compten amb dues variables creades a la funció que regulen el temps dels *delays*. La funció *dit_endavant(vel)* consta d'un *for*, que diu que quan la *i* és igual a zero i inferior al nombre assignat a la variable *maxnum*, se li suma una unitat a la variable *i*. El servomotor anirà a la posició indicada per el valor d'aquesta. La velocitat d'aquest procés és inversament proporcional a l'augment del *delay*.

La funció *dit_enrere(vel2)* és molt similar a l'anterior, ara bé, com a sentència inicial la *i* és igual al nombre establert a *maxnum* i sempre serà major que 15. Si es compleixen aquestes condicions, aleshores se li resta una unitat a la variable. La resta del procediment és igual al de la funció que li precedeix.

```
void dit_endavant(int vel)
        {
          for (int i = 0; i <= maxnum; i++)
          {
            Servo1.write(i);
            delay(vel);
          }
        }
void dit_enrere(int vel2){
        for (int i = maxnum; i >= 15; i--)
          {
            Servo1.write(i);
            delay(vel2);
          }
        }
```

Pel que fa al *void setup*, hem adjudicat un pin a cada servomotor. També hem utilitzat el *pinMode(pin, mode)* per definir els pins dels interruptors com a entrades (*inputs*).

```
void setup()
  {
    Servo1.attach(7);
    Servo3.attach(11);
    Servo4.attach(12);
    pinMode(inputPin, INPUT);
    pinMode(inputPincalibrar, INPUT);
  }
```

El programa principal o *void loop*, s'inicia col·locant el servomotor del dit a 15 graus, la posició on queda recollit dins la tapa. Segonament s'associa la variable *calibrar* amb el resultat de la funció *DigitalRead* del pin.

```
void loop() {
  Servo1.write(15);

    calibrar = digitalRead(inputPincalibrar);
   if (calibrar == HIGH)
     {
           Servo3.write(90);
           Servo4.write(90);
     }
```

Si aquest resultat és *HIGH*, és a dir, l'interruptor està emetent una senyal, els servomotors de les rodes van a la posició 90, on haurien d'estar en total repòs. En cas de que no estiguessin calibrats, girarien fins que es col·loqués el potenciòmetre en la posició correcta.

D'altra banda, tenim la part del programa que s'encarrega del moviment del nostre robot. Quan l'interruptor de la coberta s'activa, la funció *random* genera un nombre de l'1 al 10 de forma aleatòria, i depenent del resultat entra dins d'un cas o d'un altre. Dins de cada cas podem trobar les funcions mencionades al principi d'aquest apartat, però és indispensable que estiguin les funcions *dit_endavant(vel)* i *dit _enrere(vel2),* sinó el mecanisme no apagaria l'interruptor. Els nombres que apareixen dins dels parèntesis de les funcions són el valor que s'introdueix a la pròpia variable de la funció, en el nostre cas, els *delays* que volem que es realitzin.

Aquí es mostren els dos primers casos, però podreu trobar el programa complet als annexos.

```
value = digitalRead(inputPin);
 if (value == HIGH)
   {
     nombre=random(1,10);
       if (nombre==1)
         { anar_endavant(3000);
           parar();
           delay(50);
           anar_enrere(3000);
           parar();
           dit_endavant(5);
           dit_enrere(8);
           delay(1500);
         }
       else if (nombre==2)
       {
         dit_endavant(6);
         dit_enrere(5);
         delay(1500);
       }
```

3.3 Construcció

3.3.1 Eines i Materials

3.3.1.1 Eines

- Soldador
- Serra de marqueteria
- Serra de cinta
- Serra de disc
- Tornavís
- Martell
- Pinzell
- Multímetre
- Alicates
- Clau Allen
- Llima
- Pela cables
- Pistola termofusible
- Trepant
- Peu de rei digital
- Cúter
- Grapadora

Figura 10. Serra de cinta.

3.3.1.2 Materials

- Servomotor Futaba S3003
- Servomotor Feetech FS5103B
- Placa d'Arduino UNO
- Joc de cables per a electrònica
- Commutadors bipolars
- Commutador bipolar amb palanca
- Portapiles per a 4 bateries AA
- Portapiles de botó per a una pila de petaca
- Placa de prototyping/protosheet
- Adaptador d'alimentació per la placa d'Arduino
- Piles AA d'1,5V
- Pila de petaca de 9V
- Estany
- Fusta de contraplacat de 5 mm de gruix
- Pèls de serra de marqueteria
- Vernís
- Pintura negra
- Aiguarràs
- Cola de contacte
- Cola blanca
- Barres de cola termofusible
- Rodes de 360º
- Cinta aïllant
- Vetes adherents
- Resistència de 10 ± 0,5 kΩ
- Frontisses
- Rodes per servomotors

Figura 11. Servomotor Futaba S3003.

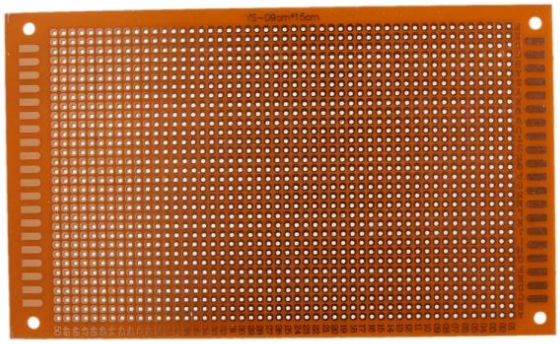


Figura 12. Placa de prototyping.

3.3.2 Procés constructiu

Un cop teníem les peces dissenyades, vam portar-les a un fuster per a que fes un tall més recte de les fustes, a més de fer una ranura a les cantonades per a que encaixessin bé les peces. Vam encolar les dues peces que formarien la tapa i les vam deixar un dia per a que s'acabessin d'adherir bé. Aleshores vam dibuixar al centre un cercle de 10 mm de diàmetre i amb un trepant el vàrem foradar. A continuació, per ampliar el forat, vam llimar-lo amb un paper de vidre col·locat de forma cilíndrica. Vam introduir la palanca de l'interruptor i el vam enganxar amb cola termofusible a l'interior de la tapa. Després vam tornar a dur aquesta peça al fuster per a que fes un tall en paral·lel a 160 mm del costat, amb un angle de 45º respecte la base.

A la base vam dibuixar el quadrat de 60 mm de costat i els rectangles que anaven enganxats a aquest, de 90x20 mm. Amb el trepant vàrem fer un forat per a poder passar el pèl de la serra i començàrem a tallar la fusta.

Un cop modificades les peces, les vam envernissar dues vegades i pintar-les de negre per l'altre costat. Aquest procés va durar 4 dies, ja que necessitaven temps per assecar-se i no es podien pintar les dues cares el mateix dia.

Mentrestant, vam desmuntar l'estructura del nostre robot sumo per aprofitar la planxa metàl·lica i la vam aplanar colpejant-la amb un martell contra una superfície llisa.

Quan la pintura de la part inferior de la base ja era seca, vam enganxar-li la planxa metàl·lica amb cola termofusible. Com que vam veure que el gruix de la fusta no era suficient i possiblement es trencaria pel pes dels servomotors vam tallar una peça de 80x70 mm i la vam enganxar a partir de la meitat de la peça metàl·lica. Seguidament, li vam donar una capa de negre per mantenir l'estètica interior.

Figura 13. Peça de fusta enganxada a la base abans de ser pintada.

Després vam encaixar les quatre parets i les vam segellar utilitzant la pistola termofusible. Ara bé, ens trobàrem amb el problema de que la base no entrava per l'excés de cola, i amb un cúter el vam retallar. Un cop la base ja estava encolada, enganxàrem al centre d'aquesta la roda mòbil per aportar més mobilitat al nostre robot.

Amb la caixa muntada el següent que vam fer va ser modificar la *protosheet*[16] proporcionada per l'AESS. Primer de tot vàrem dessoldar la connexió de l'alimentació dels servomotors, ja que utilitzaríem bateries externes. Tot seguit, vam connectar l'interruptor general, el de la coberta. Vam soldar un Pin a la placa, i vam crear un circuit d'estany fins el *GND*[17]. Aquest mateix el vam soldar a la resistència de 10KΩ i a un cable que conduïa fins un pol de l'interruptor. Vam connectar el terminal central de l'interruptor amb el Pin 2 i amb l'altre extrem de la resistència. Aquest tipus de connexió s'anomena *pull up.*

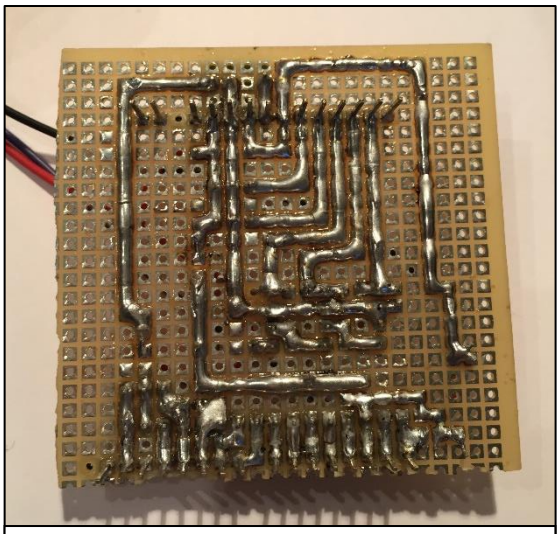


Figura 14. Circuit de la protosheet.

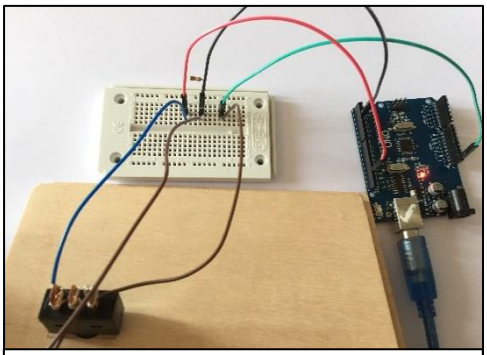


Figura 15. Connexió d'una resistència "pull up" a la placa de proves.

També vam crear un petit circuit connectant un pin amb un altre que ofereix 5V. Després, el vam associar a l'altre pol del commutador. Tot seguit, vam buscar quines instruccions s'havien d'introduir al programa per a que mostrés per pantalla si l'interruptor estava encès o apagat (*veg. programa als annexos*).

[16] Placa alternativa amb forats pre-estanyats, que serveix per crear un circuit amb diversos components, com per exemple resistències, motors o sensors.
[17] El *ground* o drenatge comú, en el context de l'electrònica, és el punt de referència de tots els senyals o un camí comú en un circuit elèctric on es poden mesurar tots els voltatges.

Tot seguit connectàrem els portapiles amb els seus respectius servomotors. Vam tallar i pelar els cables vermells dels servomotors i els soldàrem amb els vermells dels portapiles. D'altra banda, els cables negres els vam connectar tots al *GND* comú (sinó ens arriscaríem a tenir un curtcircuit).
Com que els servomotors consumeixen sempre que estan connectats, s'haurien d'anar traient les piles constantment. Per solucionar aquest problema vàrem soldar els interruptors, per obrir el circuit quan fos necessari.

A ull vàrem dissenyar una peça que faria la funció de dit. La vam serrar i posteriorment llimar. Un cop muntada la caixa ens vam trobar amb el problema de que no encaixava bé, i vam haver de crear una altra.
Amb un peu de rei digital vam mesurar la llargada de la palanca de l'interruptor i vam fer un dibuix a escala real (1:1) de la part de la caixa on aniria el dit de fusta. Vam deixar 10 mm de marge respecte el lateral, i vam marcar un punt on tocaria una part del braç.

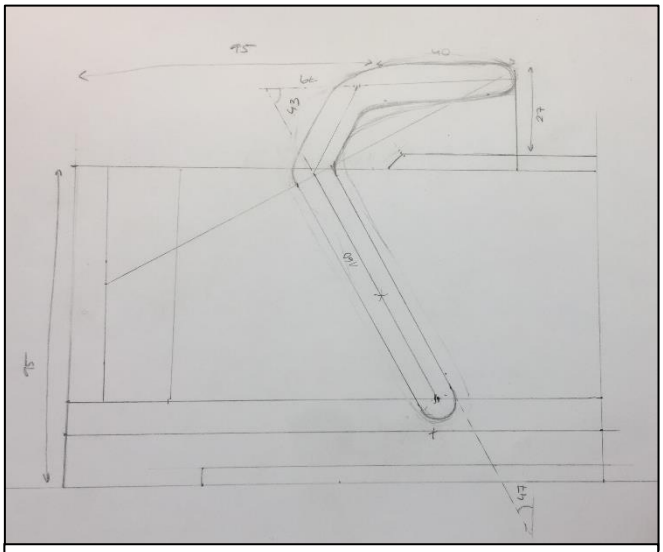


Figura 16. Dibuix a escala 1:1 de com hauria de ser el dit de fusta.

Vam unir la part superior de la palanca amb aquest punt, i fent la mediatriu vam trobar en ella el punt equidistant, on hauríem de col·locar el servomotor. Després vam dibuixar una paral·lela a cada costat de la mediatriu, separades 6 mm.
Seguidament vam acabar de fer el dibuix a mà, vigilant que l'amplada de la peça fos sempre de 12 mm.

A continuació vam col·locar el paper sobre la fusta i vam repassar la silueta de la peça amb un llapis, marcant amb força. Després vàrem resseguir el solc amb un llapis i amb la serra de cinta vam tallar la peça. Al ser corba van ser necessaris diversos talls perpendiculars a la fusta, per anar traient l'excés de fusta. Després vam acabar de refinar la peça, tallant els sobrants i llimant-la.

Després vam cargolar el dit a la peça allargada i simètrica que encaixa amb l'eix del motor, havent unit aquesta prèviament al servo.
Vam recobrir el dit amb una capa de tela amb pèl de color negre enganxat. Amb cola termofusible i grapes, vam fer la funda del dit. La dissenyàrem per a que fos extraïble, així si no quedava tal i com esperàvem, la podríem canviar per una altra o deixar el mecanisme de fusta. Tot seguit retallàrem els pels del teixit ja que eren molt llargs, i així el volum del dit va disminuir i s'assemblava més a una pota de gat. Després vam encaixar la peça de plàstic amb el servomotor.

Figura 17. Dit de fusta amb la funda de pèl.

Amb la serra de cinta també vam tallar un petit rectangle de fusta, de 40x20 mm, el qual vam pintar amb esprai negre. Després el vam enganxar al servomotor i aquest, a la base de la caixa.
També vam allargar els cables vermells soldant-los un altre cop i vam situar-los tots sota la part de la tapa que s'obria, i els vam enganxar a les parets amb cinta aïllant.

Un cop ho vam aconseguir, vam dibuixar en un tros de fusta un quadrat de 75 mm de costat i després la vam serrar, llimar i pintar. En aquesta peça vam cargolar l'Arduino amb uns cargols de 5 mm de longitud. La placa ja disposava de forats per a poder-la fixar. Tot seguit, vam enganxar unes tires de Velcro a la part posterior de la fusta i la vam enganxar a la paret interior de la caixa.

Després vam connectar els terminals laterals de l'interruptor a la corrent i el central al pin 5, per poder calibrar els servomotors sense haver d'introduir un programa nou cada cop que ho necessitéssim. Per assegurar-nos de que el commutador funcionava, vam utilitzar el mateix programa que per l'interruptor de la tapa.

Començàrem el període de proves. Vam haver d'anar provant de canviar els graus del servomotor per a que encaixés bé a la caixa. Després de diversos experiments i fracassos (*veg. apartat 5*), finalment vam veure era necessari que no passés els 15º respecte la base.

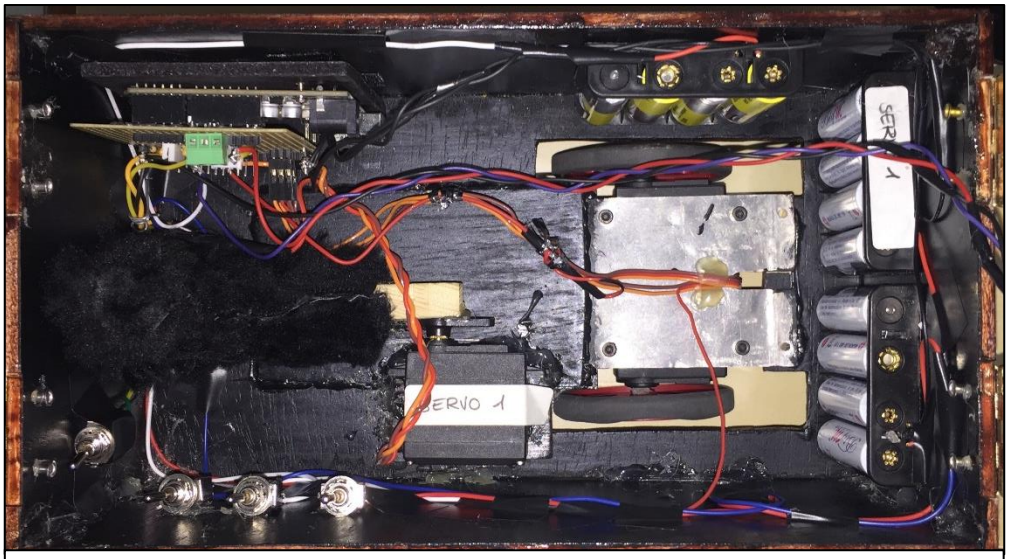


Figura 18. Connexions a l'interior de la caixa.

Només calia enganxar la tapa. Per fer-ho, vam utilitzar frontisses i vam enganxar a una pestanya per dins amb termofusible i l'altre, amb l'articulació per fora, la vam cargolar. A continuació, tal com s'explica a l'apartat 5, vam haver de rebaixar la fusta on les pestanyes de les frontisses feien contacte amb l'interior de la caixa. Amb un cúter vam extreure la major part de la fusta que necessitàvem, i amb una llima vam acabar de donar-li el toc final. Després vam tornar a pintar i envernissar la part interior de la caixa que havíem modificat.

Figura 19. Frontisses cargolades a la caixa.

Finalment, vam carregar el programa a la placa i vam accionar els interruptors de tots els servomotors, col·locats sota la tapa. Vam comprovar que el robot funcionés, i un cop vam veure que havíem assolit el nostre objectiu, vam donar la part pràctica per acabada.

Per utilitzar la caixa inútil el primer que s'ha de fer és accionar l'interruptor que tanca el circuit d'alimentació de la placa d'Arduino, situat a l'interior de la caixa. Tot seguit, també s'han d'accionar els tres interruptors que connectats als servomotors. Després s'acciona que està separat dels tres anteriors, i si les rodes giren significa que els servomotors estan desajustats. Per solucionar-ho, s'ha d'introduir un tornavís molt fi al forat lateral dels motors, i fer-lo girar fins que les rodes parin de moure's. A continuació, s'ha de desactivar el commutador bipolar i la caixa ja està preparada per el seu ús.

Figura 20. Caixa acabada.

A l'apartat 5 dels annexos hem adjuntat un enllaç on hi ha el vídeo "Construint el nostre robot".

4. Problemes i solucions

Durant la construcció del robot hem ensopegat amb diversos problemes i per cadascun d'ells hem hagut de trobar una solució:

- **La dificultat inicial del curs de Introducció a la programació en C**

La dificultat amb la que va començar el curs ens va sorprendre a tots, l'única solució possible era practicar més a casa.

- **El programa del robot de sumo no ens va funcionar com ho hauria d'haver fet**

L'últim dia del curs d'Introducció a la Robòtica, el dia de l'entrega dels robots i dels combats entre aquests, el nostre programa funcionava perfectament, però vam intentar millorar-lo i aleshores va fallar. Finalment, vam haver d'utilitzar un programa bàsic alternatiu que ens van proporcionar els nostres professors.

- **Discrepàncies entre nosaltres a l'hora de decidir quin tipus de robot faríem**

Un cop escollit que faríem una *caixa inútil*, hi havia una notable diferència d'opinió. S'havia de decidir entre construir-la amb més d'un commutador o construir-ne una igualment complexa però amb un únic interruptor. Finalment ens decantàrem per la segona opció.

- **El canvi de llenguatge C a Arduino**

L'única solució factible era aprendre la programació en Arduino de forma autodidacta, per sort, Internet va ser de gran ajuda.

- **El tipus de fusta i el gruix d'aquesta**

Principalment tots dos vam pensar en un tipus de fusta massissa i lleugera, a la vegada que fàcil de tallar i manejar. No volíem arriscar-nos a que el robot no tingués el parell motor suficient per moure la fusta, per tant, ens vam decantar per un contraxapat de 5 mm de gruix.

- **El fet de que el robot portés rodes**

Al principi tots dos creiem que el robot havia de portar rodes, però mes tard, per la falta de temps, ens vam fer enrere. Finalment, acceptant que hauríem de treballar més, vam decidir posar-li rodes. Per aportar estabilitat a la caixa, enganxàrem una roda mòbil que s'adaptava perfectament a les mesures requerides.

- **Com aconseguir que no es veiessin les rodes**

Com hem explicat anteriorment, la idea del fals fons no només va solucionar el problema de com fer que la fusta aguanti el robot per a que aquest pogués dur rodes, sinó que també deixava amagades les rodes simulant que no en porta solucionant també aquest problema.

- **Quin tipus de roda col·locar per aportar estabilitat**

Vam debatre si la roda del davant havia de ser unidireccional o de conduida. Vam trobar un joc de quatre rodes una mica més grans del que teníem previst però que encaixaven perfectament amb l'altura del robot. Posteriorment vam haver de bloquejar la direcció d'aquesta roda, però deixant la possibilitat de desbloquejar-la per si en un futur volíem fer que el robot girés.

- **Com programar 3 servomotors alhora i com alimentar-los**

Aquest dubte va romandre diverses setmanes als nostres caps, però vam trobar una solució: utilitzar una *protosheet* per connectar-los a la placa i fer servir una font d'alimentació externa.

- **Com alimentar la placa d'Arduino**

Per fer les proves amb la *protoboard* era suficient amb alimentar la placa des del cable USB que la connecta amb l'ordinador. Però necessitàvem que el robot final tingués autonomia, i vam comprar un connector amb passa-cables de 5,5x2,1 mm. Aquest el vàrem soldar a un portapiles de botó, on s'encaixava una pila de petaca de 9V.

- **Els servomotors de les rodes es desajustaven constantment**

Els motors que vam utilitzar en un principi eren els que vàrem trucar nosaltres per al robot de sumo, però aquests es desajustaven tota l'estona probablement perquè la pestanya de fi de cursa no estaria ben llimada. L'AESS ens va proveir dos servomotors de rotació contínua (360º) que es calibren des dels laterals. Això també va solucionar el problema de **com posar els servomotors de manera que es poguessin calibrar sense haver de desmuntar la caixa.**

- **Les rodes no encaixaven a la base de la fusta**

Com que vam canviar de costat els servomotors, els eixos de rotació també van variar, per tant les rodes no quadraven del tot bé amb el forat de la base. Simplement vam haver d'ampliar el forat de la base.

- **Al haver fet més gran el forat de les rodes el pes dels motors no podia ser suportat per el tros de fusta**

Per reforçar la fusta vam serrar un tros més de contraplacat i el vàrem enganxar amb cola termofusible a la part interna de la base.

- **Els sensors no detectaven bé la presència**

Primer començàrem a fer proves amb els sensor d'ultras per veure si podien detectar un dit. Al ser una superfície petita i corba les dades proporcionades eren molt irregulars. Després vam utilitzar el circuit de la *protosheet* per provar els sensors d'infrarojos i les dades van millorar, però no tant com per solucionar el problema.

- **El dit mecànic estava enganxat al servomotor 40º desfasat**

Quan vam cargolar el dit mecànic al servomotor, no ho vam fer col·locant-lo horitzontal a la posició zero. Això va suposar un problema, ja que el dit no quedava recollit dins la caixa i el programa no acceptava la posició -40º. Per això vam haver de desenganxar el servomotor de la base de la caixa. Vam escalfar un ganivet per a poder fondre la cola termofusible. Un cop extret, vam haver de descargolar el servomotor del dit mecànic. En aquest moment ens trobàrem amb

un altre problema: la peça de fusta es va trencar. Aprofitant això, descargolàrem la peça de plàstic del servomotor, mentre aquesta romania enganxada a la fusta. Després vam arreglar la fusta i vam enganxar el servomotor on estava prèviament. El dit va quedar com a una peça extraïble del servomotor, així es pot ajustar als graus que volem.

- **La col·locació de les frontisses**

Aquest va ser el problema que més ens va costar resoldre. Com que la tapa estava bisellada, vam haver de rebaixar la fusta per a poder enganxar les frontisses per dins. No trobàvem com col·locar-les de forma que no quedessin espais, és a dir, quan s'enganxaven les dues parts de la frontissa a la part de dins, la tapa quedava aixecada o més avançada. El que vam haver de fer va ser posar una part metàl·lica dins i l'altra, amb l'articulació, a la part exterior.
També vam haver de rebaixar una mica la fusta a la part lateral inferior de la caixa, on coincidien les frontisses, per a que així la tapa no quedés aixecada.

- **La caixa estava plena de pols i els cables molt enredats**

Després d'haver llimat les parts rebaixades, la caixa va acabar plena de pols. Els cables ja estaven força enredats i no van facilitar la neteja. Per tant, vam desfer les soldadures, netejar la capsa, organitzar els cables i tornar a soldar-los, en aquest ordre.

- **Un dels servomotors proporcionats per l'AESS es descalibrava constantment**

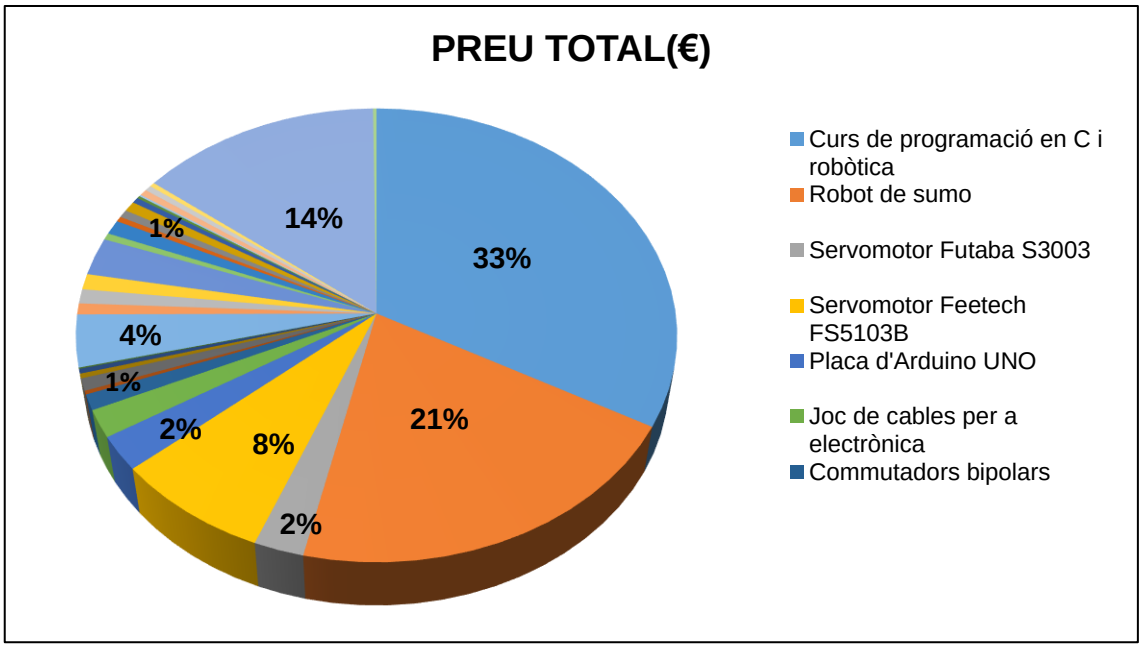
Com a possible solució vam pensar en col·locar-hi relés, interruptors digitals, de manera que quan el robot acabés de moure's, s'obrís el circuit i els servomotors paressin de rebre alimentació. No obstant, no disposàvem del temps suficient per intentar-ho, ja que no ens volíem arriscar a que la caixa deixés de funcionar. Vam tenir una idea tan senzilla com efectiva: vam fixar el tornavís a la ranura del potenciòmetre, fent així que aquest no es mogués i el motor no es desajustés.

5. Pressupost

PRODUCTE	PREU(€)	UNITATS	PREU TOTAL(€)
Curs de programació en C i robòtica	80	2	160
Robot de sumo	100	1	100
Servomotor Futaba S3003	12	1	12
Servomotor Feetech FS5103B	18,9	2	37,8
Placa d'Arduino UNO	12	1	12
Joc de cables per a electrònica	9,5	1	9,5
Commutadors bipolars	1,35	4	5,4
Commutador bipolar amb palanca	1,25	1	1,25
Portapiles per a 4 bateríes AA	1,4	3	4,2
Portapiles de batería de botó	1,6	1	1,6
Peça de prototyping/protosheet	1,6	1	1,6
Adaptador alimentació cable placa arduino	0,45	1	0,45
Piles AA 1,5V	4,5	4	18
Pila de petaca 9V	1,9	2	3,8
Estany	4,95	1	4,95
Fusta de contraplacat de 5mm de gruix	2,69	2	5,38
Tall de la fusta	13	1	13
Pèls de serra de mercateria	2,3	1	2,3
Vernís	4,95	1	4,95
Pinzell	0,9	2	1,8
Pintura negra	2,7	1	2,7
Aiguarràs	3,5	1	3,5
Cola de contacte transparent	1,95	1	1,95
Fulles de vidre	0,9	1	0,9
Cola termofusible	0,2	3	0,6
Rodes de 360º	2,15	1	2,15
Cinta aillant	1,9	1	1,9
70 cm de veta adhesiva	0,75	2	1,5
T10	9,9	7	69,3
Frontisses	0,5	2	1
TOTAL			485,48

El nostre projecte ha costat en total 485,48€. Els cursos de programació i robòtica representen la major part del preu total (33%), seguits pel robot de sumo que vam necessitar pel curs (21%). A continuació, les T10 que vam utilitzar pel transport (14%), els servomotors (10%), el joc de cables (2%) i els commutadors bipolars (2%).

Tal i com es mostra al gràfic, la placa d'Arduino UNO representa un 4% del preu total, tot i que nosaltres no la vam haver de pagar, ja que l'AESS ens la va proporcionar una com a premi del concurs de disfresses de robots. El 14% restant es divideix entre la resta del pressupost, ja que els percentatges eren molt baixos (0 o 1%) com per introduir-los al gràfic.



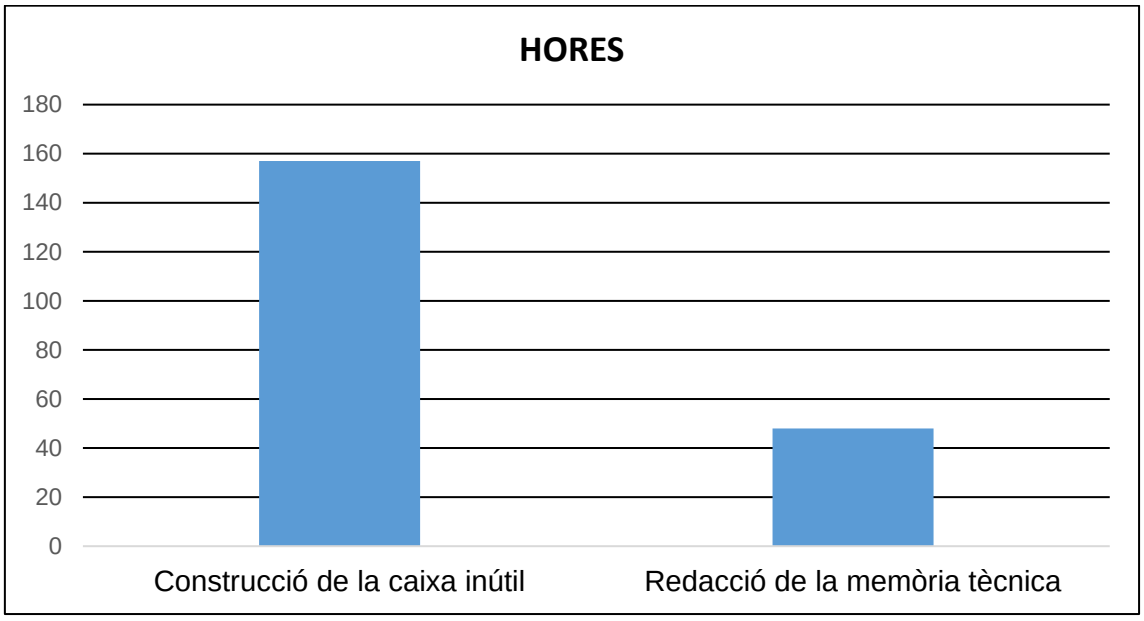
6. Hores de treball

Per realitzar aquest treball hem invertit aproximadament 214 hores cadascun. Dins la graella incloem el curset de programació en C i el de robòtica, a més de les hores de transport diàries.

DATA	Nº D'HORES	TREBALL REALITZAT
Del 26 al 30 de Juny del 2017	30	Curs d'introducció a la programació en C
Del 10 al 15 de Juliol i del 18 al 23 de Juliol del 2017	60	Curs d'introducció a la robòtica
24 de Juliol del 2017	3	Decisió del projecte
25 de Juliol del 2017	4	Començament del programa en el llenguatge C.
4 d'Agost del 2017	3	Canvi del llenguatge de programació i modificació del projecte (a l'AESS)
11 d'Agost del 2017	5	Continuació del programa (a l'AESS)
12 d'Agost del 2017	3	Decisió del disseny de la caixa (a l'AESS)
30 d'Agost del 2017	4	Disseny de la caixa en funció de l'estructura del robot de sumo (a l'AESS)
15 de Setembre del 2017	3	Recopilació de les despeses gastades fins el moment
19 de Setembre del 2017	3	Compra del servomotor i les fustes
27 de Setembre del 2017	6	Tall de les fustes, començament de pintura i vernís d'aquestes i construcció del primer "dit" de fusta
28 de Setembre del 2017	3	Millora del programa (a l'AESS)

13 d'Octubre del 2017	3	Finalització de la construcció de la base, acoblament de l'interruptor a la coberta de la caixa i finalització de la primera capa de vernís
28 d'Octubre del 2017	3	Realització de la segona capa de vernís i acoblament de les parets de la caixa.
29 d'Octubre del 2017	3	Dessoldadura de la protosheet i soldadura d'altres components i acoblament total de la caixa
2 de Novembre del 2017	2	Prova de servomotors per separat i amb diferents programes (a l'AESS)
21 de Novembre del 2017	2	Soldadura de les piles i segona prova dels servomotors
24 de Novembre del 2017	3	Solució al problema dels servomotors: no giraven a la mateixa velocitat, i, de vegades, tampoc en el mateix sentit (a l'AESS)
13 de Desembre del 2017	2	Soldadura de tots els components electrònics (a l'AESS)
15 de Desembre del 2017	2	Solució de problemes del programa i prova d'aquest en el robot
17 de Desembre del 2017	3	Començament de la part escrita
13 de Gener del 2018	5	Avançament de la memòria escrita
14 de Gener del 2018	5	Avançament de la memòria escrita
17 de Gener del 2018	4	Millorar les connexions entre els servomotors, solucionar el problema del dit mecànic i modificar el programa
20 de Gener del 2018	4	Avançament de la memòria escrita
21 de Gener del 2018	6	Avançament de la memòria escrita

22 de Gener del 2018	3	Avançament de la memòria escrita
23 de Gener del 2018	2	Col·locació de les frontisses
24 de Gener del 2018	4	Resoldadura de les connexions i segona capa de pintura interior
26 de Gener del 2018	4	Avançament de la memòria escrita
27 de Gener del 2018	5	Avançament de la memòria escrita
28 de Gener del 2018	5	Avançament de la memòria escrita
30 de Gener del 2018	3	Correcció i millora de la memòria escrita
1 de Febrer del 2018	3	Correcció i millora de la memòria
2 de Febrer del 2018	2	Correcció i millora de la memòria escrita
3 de Febrer del 2018	5	Realització dels plànols amb AutoSketch
4 de Febrer del 2018	4	Correcció i millora de la memòria escrita
TOTAL	212	



Com es pot apreciar al gràfic, hem dedicat molt més temps a la construcció de la nostra *caixa inútil* (73%) que a la realització de la memòria escrita (27%).

Als annexos adjuntem un diari amb les activitats que vam realitzar cada dia durant les tres setmanes que van durar els cursets a la Universitat Politècnica de Catalunya.

7. Conclusió

Per començar, estem molt satisfets amb la feina realitzada per portar a terme aquest treball de recerca. Hem hagut d'emprar moltes hores, tant per la part pràctica com per la memòria escrita. Ens ha servit per millorar la nostra organització a l'hora de treballar i ens ha obligat a seguir un ritme constant i una bona dinàmica d'equip.

Hem après moltíssim sobre robòtica, tenint en compte que ningú de nosaltres havia fet mai res semblant. Dissenyar el nostre propi robot ha sigut un repte molt gran: haver de crear el nostre programa, dissenyar la caixa, crear les connexions i soldar-les, ha fet que el nostre treball vagi més enllà dels coneixements que vam adquirir durant els cursos a l'estiu.

El nostre primer objectiu va ser aprendre a programar en C, i tot i que al principi va ser molt difícil pels dos, finalment ho vam aconseguir. Fins i tot vam aprendre un altre llenguatge de programació, l'Arduino. Trobar informació ha estat relativament fàcil gràcies a l'ajuda de tots els companys de la UPC i a l'abundant informació disponible a Internet.
Durant les dues setmanes del curset de robòtica també vam aconseguir el segon objectiu inicial, crear un robot lluitador de sumo. A més, vam guanyar el concurs de disfresses de robots i algunes batalles dins del ring.

El projecte final ha patit algunes modificacions respecte la proposta inicial, que hem hagut de realitzar a mesura que avançàvem en el projecte i ens trobàvem amb dificultats. No obstant, hem aconseguit crear una *caixa inútil* funcional i amb notables diferències respecte els models originals i comercials: li hem afegit rodes, d'aquesta manera es pot moure, i una funció aleatòria que fa que cada cop que s'acciona l'interruptor la resposta sigui diferent respecte l'anterior (hi ha deu casos possibles).
A causa de la falta de temps i del desconeixement sobre el tema, no hem estat capaços de posar-li sensors al robot per a que detectessin un dit humà. Cal dir que podríem haver alimentat la placa amb una sola bateria, sense necessitar quatre portapiles.

En quant al disseny de la caixa, encara que té un bon aspecte envernissat, es podria millorar encaixant millor les peces. També ens ha faltat més moderació a l'hora de col·locar la cola termofusible i més ordre pel que fa a les connexions cablejades.

En conclusió, hem solucionat tots els contratemps que ens han sorgit i hem aconseguit construir una *caixa inútil* que apaga l'interruptor en sentit contrari al que s'acciona, i que cada cop executa un cas diferent. Com ja hem dit, tots dos coincidim en que el robot podria haver estat més complex i més ben acabat. No obstant, estem molt contents amb el resultat obtingut.

8. Bibliografia i Webgrafia

Bibliografia

FITGERALZ, Scott. *Arduino projects book.* (Pàgines 6, 7, 8, 9 i 11)

També hem extret informació de documents proporcionats per l'AESS.

Webgrafia

1. http://arduino.cl/introduccion-a-los-tipos-de-dato-con-arduino/
2. http://diwo.bq.com/variables-en-arduino/
3. http://isaepenoinformatica.blogspot.com.es/2014/05/ventajas-y-desventajas de-lenguaje-c.html
4. http://manueldelgadocrespo.blogspot.com.es/p/operadores-compuestos.html
5. http://placarduino.weebly.com/
6. http://tecnotec.es/Auno/conexiones.html
7. http://www.dc.fi.udc.es/~so-grado/current/Varios/CursoC.pdf
8. http://www.lcc.uma.es/~janto/ftp/fundinf/trans_t4.pdf
9. https://aprendiendoarduino.wordpress.com/category/operadores/
10. https://aprendiendoarduino.wordpress.com/category/operadores/
11. https://aprendiendoarduino.wordpress.com/tag/estructuras-de-control/
12. https://programarfacil.com/blog/arduino-blog/instalar-una-libreria-de-arduino/
13. https://store.arduino.cc/arduino-uno-rev3
14. https://www.acamica.com/comunidad/6407/la-importancia-de-la-programacion-y-las-caracteristicas-de-arduino
15. https://www.aprendiendoarduino.com/tag/librerias-arduino/
16. https://www.arduino.cc/reference/es/language/structure/sketch/setup/
17. https://www.nextiafenix.com/arduino-vs-pic-la-gran-batalla/
18. https://www.youtube.com/watch?v=ERL0r4GDzqk
19. https://www.zonamaker.com/arduino/intro-arduino/conociendo-arduino-introduccion

Font de les figures

Figura 1: Font pròpia.

Figura 2: Font pròpia.

Figura 3: Font pròpia.

Figura 4: Font pròpia.

Figura 9: https://musicologia.wordpress.com

Figura 10: https://toolsnhome.com

Figura 11: https://modelarina.cz

Figura 12: https://alexnld.com

Figura 13: Font pròpia.

Figura 14: Font pròpia.

Figura 15: Font pròpia.

Figura 16: Font pròpia.

Figura 17: Font pròpia.

Figura 18: Font pròpia.

Figura 19: Font pròpia.

Figura 20: Font pròpia.

Annexos

1. Diari dels cursets d'estiu

Dia 1 - 26/06/2017

Ferran

Aquest primer dia l'he començat amb moltes ganes igual que els meus companys. La Roser m'ha caigut molt bé, però no he entès res del que ens ha explicat. Crec que han sigut masses coses el primer dia, a casa revisaré tot el que he apuntat i intentaré quedar-me amb alguna funció més.

Marina

Avui m'he despertat amb moltes ganes de començar el curs, ja que no tinc ni idea de programació. Només arribar ja han donat per suposat que teníem coneixements de codi binari i hexadecimal, i la veritat és que mai n'havia sentit a parlar del segon. Ens han explicat que són les variables i les constants i hem après a definir-les.

També hem après tots els operadors que hi ha (lògics, aritmètics i racionals). He après que hi ha un operador aritmètic que s'anomena mòdul, que es queda només amb el residu d'una divisió. Es pot utilitzar en una funció per saber si un nombre és parell.

Dia 2 - 27/06/2017

Ferran

Avui les coses han millorat i alguna cosa he pogut captar, tinc molts dubtes i el cap em dona voltes, personalment crec que està explicant masses coses en molt poc temps. Cada vegada que explica alguna cosa i m'ho apunto se'm escapa l'explicació següent, crec que deixaré d'apuntar i em centraré únicament a atendre a classe.

Marina

Aquest matí ens han introduït als controls de flux, que són molt similars a les taules de veritat de filosofia. També hem après com funciona el "switch" i com introduir algun comentari al programa. A més, hem creat el nostre primer programa en paper. Aleshores, ens han introduït a les funcions. Ens han explicat

com fer que retornin el que nosaltres vulguem i com cridar-les dins del programa principal.

Dia 3 - 28/06/2017

Ferran

Cada dia em quedo amb més funcions, ja entenc les bàsiques el "if", el "while", una mica el "switch"... Avui si que he après moltes coses, crec que és perquè ahir me'n vaig anar a dormir aviat ja que m'he adonat de que els altres dies la son s'ha estat apoderant de mi durant les explicacions de la Roser. Un company que he fa el curs amb nosaltres m'ha ajudat a fer els problemes d'avui i he fet la major part bé.

Marina

Avui hem fet tres problemes i la majoria m'han sortit. Un era de comptar les paparres que té un gat. He sortit a la pissarra a fer-lo i era correcte. Sempre hi ha algun error a cada programa però a poc a poc vaig aprenent. Crec que és molta matèria per fer en una setmana.

Al arribar a casa he repassat tot el que hem fet des del dilluns amb el meu pare que és programador. Gràcies a ell he entès encara millor les sentències iteratives i en quins casos s'han d'utilitzar.

Dia 4 – 29/06/017

Ferran

Per fi, avui he fet el primer programa sol ben fet, i ja domino el "switch"! M'he animat i ara a casa em posaré a fer algun altre programa simple. Encara m'he d'instal·lar els programes penjats al moodle, però no crec que tardi gaire en fer-ho. En quant a la classe hi ha molta diversitat d'actituds de cara al curs: estem els que el nostre Treball de Recerca depèn dels coneixements adquirits durant aquest curs i els que s'han apuntat voluntàriament al curs.

Marina

Avui ens han explicat què són les llibreries, per a què serveixen i com incloure-les al programa. També ens han ensenyat els punters i els vectors. Tot i que entenc la part teòrica del curs, segueixo tenint força dificultat pel que fa a crear programes des de 0. He fet un programa que mostri per pantalla si el nombre introduït és parell o imparell. Per poder veure-ho, hem après a utilitzar el CMD (símbol del sistema).

Dia 5 – 30/06/2017

Ferran

Avui no ha estat un gran dia en el que es refereix al curs ja que només la meitat dels programes que he intentat fer m'han sortit. Quan penso en tot el que hem après aquesta setmana m'entra un mareig desconcertant, però segueixo pensant que els coneixements adquirits són insuficients per a fer un robot, sincerament crec que ha explicat moltes coses però que li han faltat explicar moltes més, però amb una sola setmana és literalment impossible explicar més coses de les que la Roser ha explicat, es podria dir que ha sabut aprofitar molt bé el temps. Ara toca practicar a casa fent programes.

Marina

Avui ha sigut l'últim dia de curs i no hem fet gaire cosa. Simplement la professora ens ha donat una llista amb problemes i després els hem corregit. Crec que és molt difícil aprendre a programar en només una setmana, i molt menys sense haver fet res semblant abans. Tot i així, crec que aquesta setmana me n'he sortit bastant bé.

Dia 6 – 10/07/2017

Ferran

Avui hem començat el curs de Robòtica i realment m'ha donat bona impressió ja que a diferencia del curs de programació hi ha més gent apuntada apart de nosaltres i tres noies més. M'ha semblat molt preparat ja que apart de la Roser hi ha un noi que ens ajuda, el Christian. El nostre escriptori està ple de màquines rares que m'encantaria saber per a que serveixen. Entre la Marina i jo hem intentat fer un programa amb 8 LEDs que simulés les llums del cotxe fantàstic, no l'hem acabat i per el poc que portem hem fet un munt d'errors, espero solucionar-los tots demà.

Marina

Avui ha sigut el primer dia del curs de Robòtica i l'he començat preocupada perquè encara no sé programar del tot. Tots els altres que participaven, ja tenien coneixements previs de programació en C i feien tots els exercicis molt ràpid. En canvi, el meu company i jo no hem pogut acabar ni un.

Dia 7 – 11/07/2017

Ferran

El programa de la simulació dels llums del cotxe fantàstic segueix sense funcionar, el problema és que no ens dona cap error però després el simulador fa una cosa molt diferent de la que nosaltres volem que faci. Els 8 LEDs s'han d'engegar seguits i apagar-se quan el següent s'encén, no encendre's aleatòriament com fa el simulador. És molt frustrant, igual no hauria d'haver escollit robòtica com a tema de treball.

Marina

Aquest matí m'he replantejat el fet d'haver triat aquest treball de recerca, ja que després de quatre hores intentant que ens funcioni el programa no hem aconseguit res. A més, som els únics que sempre estem preguntant als professors i demanant ajuda. Hem intentat fer un programa amb vuit LEDs que simula les llums del cotxe fantàstic. Quan he arribat a casa he seguit pensant com el podríem fer però encara no veig cap manera de fer-lo.

Dia 8 – 12/07/2017

Ferran

Avui no només hem aconseguit que ens funcionés el programa del cotxe fantàstic sinó que a sobre hem aconseguit que es parés quan nosaltres vulguem. La Roser ens ha explicat com fer el programa del comptador, té bona pinta però és més difícil que el del cotxe fantàstic, toca posar-se les piles.

Marina

Avui hem aconseguit que funcioni el programa del cotxe fantàstic. Cal dir que hem picat tot el codi a mà, i que no és tan senzill ni curt com els programes dels nostres companys. El "pic-simulator" no processava bé no es podia apreciar com s'il·luminarien els LEDs, així que hem après a utilitzar la "protoboard" i ho hem provat. La llum del final s'engegava dos cops, però aquest problema s'ha solucionat ràpidament amb uns petits canvis al programa. També ens han ensenyat com funcionen els "7-segments" i ens han proposat de fer un comptador amb un, activant i desactivant els ports amb codi binari.

Dia 9 – 13/07/2017

Ferran

Ahir per la tarda vaig estar escrivint quins bits s'han d'activar del port D del microxip per a que surti cada número. Això avui ha estat molt útil ja que ho hem necessitat a l'hora de fer el programa del comptador, ja que per definir cada número es necessita el valor binari dels ports que ha d'activar. El programa ens ha sortit però encara té alguns problemes, demà a primera hora els solucionarem ja que avui no ens ha donat temps per fer-ho.

Marina

Avui hem aconseguit que funcioni el programa del comptador però hi ha alguns matisos que cal millorar, com que canviï cada segon i compti a temps real. També hem après a multiplexar. Com que el microxip processa molt ràpid, per no utilitzar tants ports, només engeguem una de les dues pantalles però aquestes s'alternen tan ràpidament que sembla que les dues estiguin enceses alhora.

Dia 10 – 14/07/2017

Ferran

Avui no només hem solucionat els errors del programa del comptador, sinó que també hem fet bona part d'un programa per a un comptador de dos xifres. Ja hem definit les constants, declarat les variables i decidit quines funcions utilitzarem per a fer el programa. Intentaré acabar-lo a casa i després d'instal·lar-me tots els programes necessaris per fer-lo: el simulador, el que escriu i compila el programa i el que el transmet a la placa base.

Marina

Avui hem acabat el comptador normal i després hem començat a posar en pràctica la multiplexació creant un programa pel comptador de dues xifres. Ha costat molt i hem necessitat molta ajuda perquè tot i que en el simulador funcionés bé el programa, els nombres canviaven tan ràpid als 7-segments que semblava que hi haguessin dos vuits tota l'estona.

Dia 11 – 17/07/17

Ferran

Avui hem començat la tercera i última setmana de cursos de la UPC obligatoris i necessaris per el nostre TR. Hem acabat el programa del comptador de dos xifres i ens ha funcionat perfectament, ara el repte és que pari quan nosaltres li ordenem i comenci a comptar enrere en el moment que nosaltres vulguem. No ens ha donat temps de intentar-ho, que pari ho veig bastant fàcil, posem un "if" dins del "while" amb la condició de que entri quan només una nova variable estigui activada i ja està, el problema el veig en que conti enrere, la Marina diu que te una idea de com fer-ho i demà me l'explicarà.

Marina

Al final ens hem sortit amb el comptador de dues xifres i no només hem aconseguit que funcionés, sinó que també anés a temps real. Les quatre hores diàries es fan curtes per combinar l'explicació dels professors amb els programes que hem de crear per posar en pràctica els coneixements adquirits.

Dia 12 – 18/07/17

Ferran

Hem aconseguit que el comptador pari quan se li ordena però la idea de la Marina de que conti enrere no ha funcionat. Ja casi havíem donat amb la solució, però se'ns ha acabat el temps. Aquests dies estem aprenent molt però encara no hem vist cap robot ni cap motor i se'ns acaba el temps, espero que demà ens entregui les peces del robot i puguem començar a muntar-lo i a programar-lo.

Marina

Avui el meu company i jo ens hem posat nerviosos perquè som els que més endarrerits anem, en tres dies acabem el curs i encara no tenim res muntat pel que fa al robot. Avui ens han ensenyat els tipus de motors que pot portar el nostre robot i hem creat un programa per fer girar un motor de contínua i per regular la velocitat de rotació. D'altra banda volíem posar un polsador a la protoboard que detingués el comptador de dues xifres, però al final no hem tingut suficient temps.

Dia 13 – 19/07/17

Ferran

El programa del comptador de dos xifres ja funciona a la perfecció, compta endavant i enrere. Hem adaptat el programa per a que funcioni en la nostra placa i a la protoboard. Després d'estructurar tots els LEDs, el xip i d'enllaçar cada bit dels ports D i C amb cables hem compilat el programa i l'hem copiat al xip de la protoboard. El programa ens ha funcionat a la perfecció i per fi ens han explicat com funcionen els motors que utilitzarem per al robot de sumo, els servo motors.

Marina

Després de les proves del dia anterior amb els motors de contínua, avui ens han presentat els servomotors i m'ha costat bastant entendre com funcionen. La veritat és que això dels polsos electromagnètics és força estrany, i a més programar-ho és encara més difícil. El meu company no ha escoltat l'explicació i per tant no m'ha pogut ajudar a programar els servomotors.

Dia 14 – 20/07/17

Ferran

Avui ens ha ensenyat a trucar els servomotors per a que puguin donar més d'una volta i per fi ens han arribat les peces dels robots. Després de trucar els servomotors hem començat a fer el robot. Mentre jo m'ocupava de la part mecànica la Marina anava fent el programa de sumo. Encara ens queda molt per fer i aquesta tarda a casa de Marina l'acabarem de programar i li dissenyarem una disfressa original per a que guanyi el concurs de disfresses.

Al final hem decidit disfressar-lo de bitxo pelut amb una capa de pèl que hem trobat al cofre de les disfresses de la Marina i amb uns ulls que hem comprat al "Hiber Àsia" ens ha quedat un robot molt divertit i de nom li hem posat "La Bèstia".

En quan al programa no l'hem acabat ja que jo havia de baixar després a Barcelona i no em podia quedar mes estona. La Marina s'ha quedat fent-lo però suposo que com que demà tenim entre 2 i 3 hores per acabar de fer retocs les aprofitarem per acabar el programa.

Marina

Avui ens han ensenyat a trucar els servomotors. M'he posat nerviosa perquè era la primera vegada que utilitzava una serra de disc, però ha sigut més fàcil del que pensava i a més m'han quedat uns talls molt regulars. Se'ns ha trencat el

dent d'un engranatge al desmuntar-lo i ens han donat un altre servo. A continuació he començat a fer el programa mentre el meu company muntava l'estructura. Al sortir hem comprat cartolines perquè la nostra idea era disfressar al robot de "capsa de kleenex" però finalment ens hem decantat per recobrir-lo amb una capa de pèl negre i posar-li ulls. El Ferran ha marxat aviat i he hagut d'acabar jo la disfressa i acabar el programa. Vaig acabar a les 12:30 de la nit, i fins la 1 no vaig poder anar a dormir perquè fent la disfressa ho vàrem embrutar tot.

Dia 15 – 21/07/17

Ferran

Avui hem intentat perfeccionar el programa, però al trastejar-lo, hem tocat alguna cosa que ha fet que no funcionés. Per més que ho hem revisat i hem intentat canviar el que nosaltres creiem que era l'error que impedia funcionar el nostre programa, no ho hem aconseguit i hem hagut d'instal·lar el programa de sumo que la Roser havia penjat en el moodle per a situacions com la nostra. Els combats han estat divertits i La Bèstia ha guanyat la meitat dels combats. El premi del millor programa òbviament no ens l'hem endut nosaltres, però el de la millor disfressa sí. El premi ha estat: una placa base d'Arduino, molts components per a la placa, una protoboard, un clauer multi-utilitats, resistències, sensors i LEDs.

Marina

Avui hem provat el programa que vaig dissenyar ahir, i funcionava perfectament. Però per ajustar-lo per a que el robot tingués més precisió hem tocat alguna cosa al programa que ha fet que aquest deixés de funcionar tant bé i com que el vam modificar a l'original, hem perdut tot el programa que vaig fer ahir durant tantes hores. Ha sigut molt frustrant. Al final hem hagut de descarregar un programa que ha penjat la roser al moodle per poder participar a la competició de sumo. "La Bèstia", el nostre robot, ha guanyat la meitat dels combats tot gràcies a que els altres robots sortien sols del ring. Però al menys hem guanyat el concurs de disfresses i ens han regalat un kit bàsic d'Arduino.

2. Programes en C

2.1 Comptador d'una xifra

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#define ZERO 0b00111111
#define UNO 0b00000110
#define DOS 0b01011011
#define TRES 0b01001111
#define CUATRO 0b01100110
#define CINCO 0b01101101
#define SEIS 0b01111101
#define SIETE 0b00100111
#define OCHO 0b01111111
#define NUEVE 0b01101111
void main (void)
{
      set_tris_d(0x00);
      set_tris_c(0xff);
while(1)
      {
      while (input(PIN_C1))
            {
            output_d(ZERO);
            output_toggle(ZERO);
            output_b(UNO);
            output_toggle(UNO);
            output_b(DOS);
            output_toggle(DOS);
            output_b(TRES);
            output_toggle(TRES);
            output_b(CUATRO);
            output_toggle(CUATRO);
            output_b(CINCO);
            output_toggle(CINCO);
```

```
            output_b(SEIS);
            output_toggle(SEIS);
            output_b(SIETE);
            output_toggle(SIETE);
            output_b(OCHO);
            output_toggle(OCHO);
            output_b(NUEVE);
            output_toggle(NUEVE);
            }
while (input(PIN_D2))
            {
            output_b(NUEVE);
            output_toggle(NUEVE);
            output_b(OCHO);
            output_toggle(OCHO);
            output_b(SIETE);
            output_toggle(SIETE);
            output_b(SEIS);
            output_toggle(SEIS);
            output_b(CINCO);
            output_toggle(CINCO);
            output_b(CUATRO);
            output_toggle(CUATRO);
            output_b(TRES);
            output_toggle(TRES);
            output_b(DOS);
            output_toggle(DOS);
            output_b(UNO);
            output_toggle(UNO);
            output_b(ZERO);
            output_toggle(ZERO);
            }
      }
}
```

2.2 Comptador de dues xifres

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP,PUT,BROWNOUT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C0, rcv=PIN_C1)
#define ZERO 0b00111111
#define UNO 0b00000110
#define DOS 0b01011011
#define TRES 0b01001111
#define CUATRO 0b01100110
#define CINCO 0b01101101
#define SEIS 0b01111101
#define SIETE 0b00100111
#define OCHO 0b01111111
#define NUEVE 0b01101111
byte const NInts=152;
int16 C_Ints=0;
char Flag=0;
signed int x=0;
signed int y=0;
int a=0;
int
numeros[10]={ZERO,UNO,DOS,TRES,CUATRO,CINCO,SEIS,SIETE,OCHO,NUEVE};
#int_RTCC
void RTCC_isr() {
  if(C_Ints>NInts){
    if(Flag==0){
      Flag=1;
    }
    else{
      Flag=0;
    }
    C_Ints=0;
  }
  ++C_Ints;
}
```

```
void main(void){
char K;int aux;
K=Flag;
enable_interrupts(INT_RTCC);
enable_interrupts(global);
setup_counters(RTCC_INTERNAL,RTCC_DIV_128);
set_tris_d(0x00);
   while(1){
       aux=input(PIN_C2);
         if ((y<10)&&(x<10)&&(aux==1)){
            if(K==Flag){
            output_d(numeros[x]);
            output_low(PIN_C0);
                 output_high(PIN_C0);}
            else{
               output_d(numeros[y]);
               output_low(PIN_C1);
               output_high(PIN_C1);
               K=Flag;
         y++;}}
      else if (y==10){
         x++;y=0;}
      else if (x==10){
         x=0;}
         output_d(numeros[y]);
         output_low(PIN_C1);
         output_high(PIN_C1);}
   }
```

2.3 Cotxe fantàstic

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP,PUT,BROWNOUT
#use delay(clock=2000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
void main(void)
{
      int x=0;
      set_tris_d(0x00);
      set_tris_b(0xff);

   while(1)
  {
      if(x==0)
      {
            Output_bit(PIN_D0,1);x++;delay_ms(1000);
      }
      else if (x==1)
      {
            output_toggle(PIN_D0);output_bit(PIN_D1,1);x++;
            delay_ms(1000);
      }
      else if (x==2)
      {
            output_toggle(PIN_D1);output_bit(PIN_D2,1);x++;
            delay_ms(1000);
      }
      else if (x==3)
      {
            output_toggle(PIN_D2);output_bit(PIN_D3,1);x++;
            delay_ms(1000);
      }
      else if (x==4)
      {
            output_toggle(PIN_D3);output_bit(PIN_D4,1);x++;
            delay_ms(1000); }
```

```
      else if (x==5)
      {
            output_toggle(PIN_D4);output_bit(PIN_D5,1);x++;
            delay_ms(1000);
      }
      else if (x==6)
      {
            output_toggle(PIN_D5);output_bit(PIN_D6,1);x++;
            delay_ms(1000);
      }
      else if (x==7)
      {
            output_toggle(PIN_D6);output_bit(PIN_D7,1);x++;
            delay_ms(1000);
      }
      else if (x==8)
      {
            output_toggle(PIN_D7);output_bit(PIN_D6,1);x++;
            delay_ms(1000);
      }
      else if (x==9)
      {
            output_toggle(PIN_D6);output_bit(PIN_D5,1);x++;
            delay_ms(1000);
      }
      else if (x==10)
      {
            output_toggle(PIN_D5);output_bit(PIN_D4,1);x++;
            delay_ms(1000);
      }
      else if (x==11)
      {
            output_toggle(PIN_D4);output_bit(PIN_D3,1);x++;
            delay_ms(1000); }
```

```
      else if (x==11)
      {
            output_toggle(PIN_D4);output_bit(PIN_D3,1);x++;
            delay_ms(1000);
      }
      else if (x==12)
      {
            output_toggle(PIN_D3);output_bit(PIN_D2,1);x++;
            delay_ms(1000);
      }
      else if (x==13)
      {
            output_toggle(PIN_D2);output_bit(PIN_D1,1);x++;
            delay_ms(1000);
      }
      else if(x==14)
      {
      Output_toggle(PIN_D1);x=0;
      }
   }
}
```

2.4 Prova motors

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP,PUT,BROWNOUT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
char Keypress=' ';
char opcio;
int aux;
#int_rda
void serial_isr() {
    if(kbhit()){
    Keypress=getc();
    if(Keypress!=0x00){
      putchar(keypress);
      aux=0;
    }
  }
}
#INT_TIMER1
void isr_timer1(){
opcio=keypress;
switch(opcio){
   case 'a':
      switch (aux){
         case 0:
            output_high(PIN_B2);
            output_high(PIN_B1);
            set_timer1(65223);
            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 'w':
     switch (aux){
         case 0:
            output_high(PIN_B2);
            output_high(PIN_B1);
            set_timer1(65223);
            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 'c':
      switch (aux){
         case 0:
            output_high(PIN_B2);
            output_high(PIN_B1);
            set_timer1(64598);
```

```
            aux=1;
            break;
         case 1:
            output_low(PIN_B1);
            output_low(PIN_B2);
            set_timer1(53973);
            aux=0;
            break;
      }
      break;
   case 's':
      output_low(PIN_B2);
      output_low(PIN_B1);
      break;
      }
}

void main(){
   setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
   enable_interrupts(int_rda);
   enable_interrupts(global);
   enable_interrupts(INT_TIMER1);
do{
}while(1);}
```

2.5 Robot sumo

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP,PUT,BROWNOUT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
char moviment=' ';
int aux=0;
int lesquerra, lcentral, ldreta;
int uesquerra, udreta;
void linia(){   set_adc_channel(5);
   delay_ms(1);
   ldreta = read_adc();

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

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

   delay_ms(100);
   }
void ulls()
{
   set_adc_channel(2); //activem la sortida que volem llegir
   delay_ms(1);
   udreta = read_adc(); //llegim el valor

   set_adc_channel(3); //activem l'altre sortida
   delay_ms(1);
   uesquerra = read_adc(); //llegim el valor

   delay_ms(100);
}
```

```
#INT_TIMER1
void isr_timer1(){
      switch(moviment){
            case 'a': //endavant
                  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': //parar
                  switch(aux){
                        case 0:
                              output_high(PIN_B1);
                              output_high(PIN_B2);
                              set_timer1(64599);
                              aux=1;
                        break;
```

```
                        case 1:
                              output_high(PIN_B1);
                              output_low(PIN_B2);
                              set_timer1(53974);
                              aux=0;
                        break;
                  }
            break;
            case 'c': //enrere
                  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 'd': //gir dreta
                  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(53349);
                              aux=0;
                        break;
                  }
            break;
            case 'e': //gir esquerra
                  switch(aux){
                        case 0:
                              output_high(PIN_B1);
                              output_high(PIN_B2);
                              set_timer1(63973);
                              aux=1;
                        break;
                        case 1:
                              output_high(PIN_B1);
                              output_low(PIN_B2);
                              set_timer1(54598);
                              aux=0;
                        break;
                  }
            break;
      }
}
void main(){
   setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
   set_tris_b(0x00);
   setup_adc(ADC_CLOCK_INTERNAL);
   setup_adc_ports(ALL_ANALOG);
   enable_interrupts(int_rda);
   enable_interrupts(global);
```

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

      if (lesquerra<=20){
            moviment='c';
            deñau_ms(100);
      }
      else if (ldreta<=20){
            moviment='c';
            deñau_ms(100);
      }
      else if ((lesquerra<=20)&&(lcentral<20)){
            moviment='c';
            moviment='d';
      }
      else if ((lesquerra<=20)&&(lcentral<20)){
            moviment='c';
            moviment='d';
      else if ((ldreta<=20)&&(lcentral<20)){
            moviment='c';
            moviment='e';
      }
      else if ((udreta<100)&&(uesquerra<100)){
            moviment='a';
      }
      else {
            moviment='a';
            delay_ms(200);
            moviment='d';
            delay_ms(200);
            moviment='a';
            delay_ms(200);
            moviment='d';
      }
}
```

3. Programes en Arduino

3.1 Caixa inútil

```
#include <Servo.h>
int vel3 = 0;
int vel4 = 0;
int maxnum = 110;
const int inputPin = 2;
const int inputPincalibrar = 5;
int calibrar = 0;
int value = 0;
int nombre;
Servo Servo1;
Servo Servo3;
Servo Servo4;

void anar_endavant(int temps)
          {
            vel3 = 180;
            vel4 = 0;
            Servo3.write(vel3);
            Servo4.write(vel4);
            delay(temps);
          }
void parar()
          {
          vel3 = 90;
          vel4 = 90;
          Servo3.write(vel3);
          Servo4.write(vel4);
          }
void anar_enrere(int temps2)
          {
            vel3 = 0;
            vel4 = 180;
```

```
            Servo3.write(vel3);
            Servo4.write(vel4);
            delay(temps2);
          }
void dit_endavant(int vel)
        {
          for (int i = 0; i <= maxnum; i++)
          {
            Servo1.write(i);
            delay(vel);
          }
        }
void dit_enrere(int vel2){
        for (int i = maxnum; i >= 15; i--)
          {
            Servo1.write(i);
            delay(vel2);
          }
        }



void setup() {
Servo1.attach(7);
Servo3.attach(11);
Servo4.attach(12);
pinMode(inputPin, INPUT);
}
void loop() {
  while(true) {
  Servo1.write(15);

     calibrar = digitalRead(inputPincalibrar);
     value = digitalRead(inputPin);
```

```
if (calibrar == HIGH)
      {
            Servo3.write(90);
            Servo4.write(90);
      }

    else if (value == HIGH)
      {
        nombre=random(1,10);
          if (nombre==1)
            { anar_endavant(3000);
              parar();
              delay(50);
              anar_enrere(3000);
              parar();
              dit_endavant(5);
              dit_enrere(8);
              delay(1500);
            }
          else if (nombre==2)
          {
            dit_endavant(6);
            dit_enrere(5);
            delay(1500);
          }

          else if (nombre==3)
          {
            anar_enrere(1500);
            parar();
            delay(100);
            anar_endavant(1500);
            parar();
            dit_endavant(3);
            dit_enrere(8);
            delay(1500);}
```

```
          else if (nombre==4)
          {
            dit_endavant (7);
            dit_enrere(15);
            delay(1500);
          }
          else if (nombre==5)
          {
            dit_endavant (3);
            dit_enrere(3);
            delay(1500);
          }
          else if (nombre==6)
          {
            dit_endavant (10);
            dit_enrere (10);
            delay(1500);
          }
          else if (nombre==7)
          {
            dit_endavant(5);
            dit_enrere(8);
            delay(1500);
          }
          else if (nombre==8)
          {
            delay(1000);
            dit_endavant(10);
            dit_enrere(3);
            delay(1500);
          }
          else if (nombre==9)
          {
            dit_endavant(5);
            dit_enrere(8);
```

```
            delay(1500);
          }
          else
            {
            dit_endavant(10);
            dit_enrere(3);
            delay(3000);
            }
        }
    }
}
```

3.2 Prova interruptor

```
#include<Servo.h>
Servo Servo3;
Servo Servo4;
const int inputPin = 5;

int value = 0;

void setup() {
  Serial.begin(9600);
  pinMode(inputPin, INPUT);
  Servo3.attach(11);
  Servo4.attach(12);
}

void loop(){
  value = digitalRead(inputPin);  //lectura digital de pin
  if (value == HIGH) {
      Serial.println("Encendido");
  }
  else {
      Serial.println("Apagado");
  }
  delay(1000);
}
```

3.3 Prova dels servomotors trucats

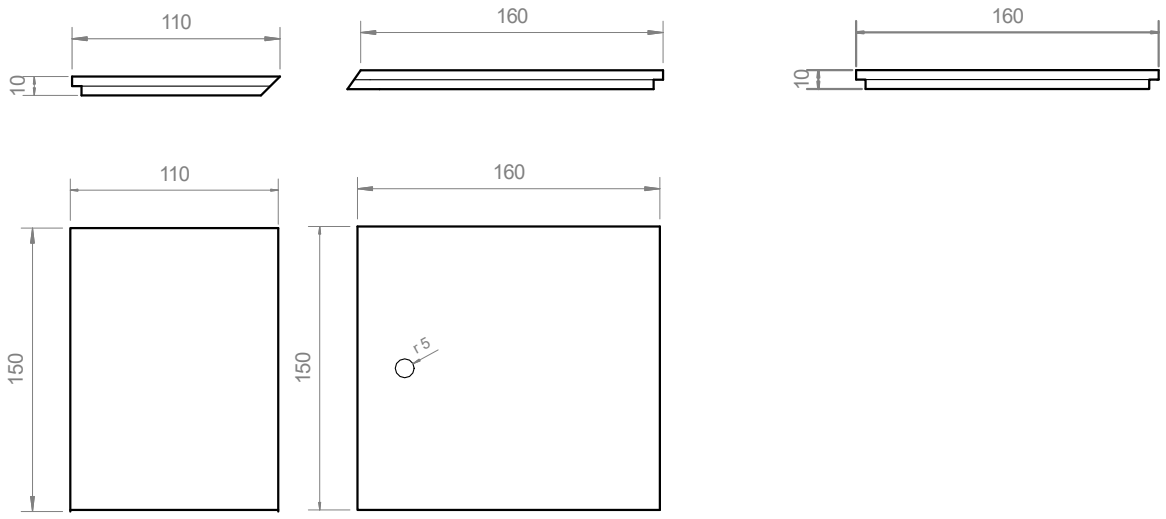
```
#include <Servo.h>
int value;
Servo servo3;
Servo servo4;
void anar_endavant(){
          servo3.write(180);
          servo4.write(-180);
          }
void parar(){
           servo3.write(90);
           servo4.write(90);
          }
void anar_enrere(){
          servo3.write(-180);
          servo4.write(180);
          }
void setup() {
      servo3.attach(5);
      servo4.attach(6);
}
void loop() {
      value = digitalRead(2);
       if (value == HIGH){
            anar_endavant();
            delay (3000);
            anar_enrere();
            delay (3000);
            parar();
            delay (3000);
      }
}
```

3.4 Prova servomotors

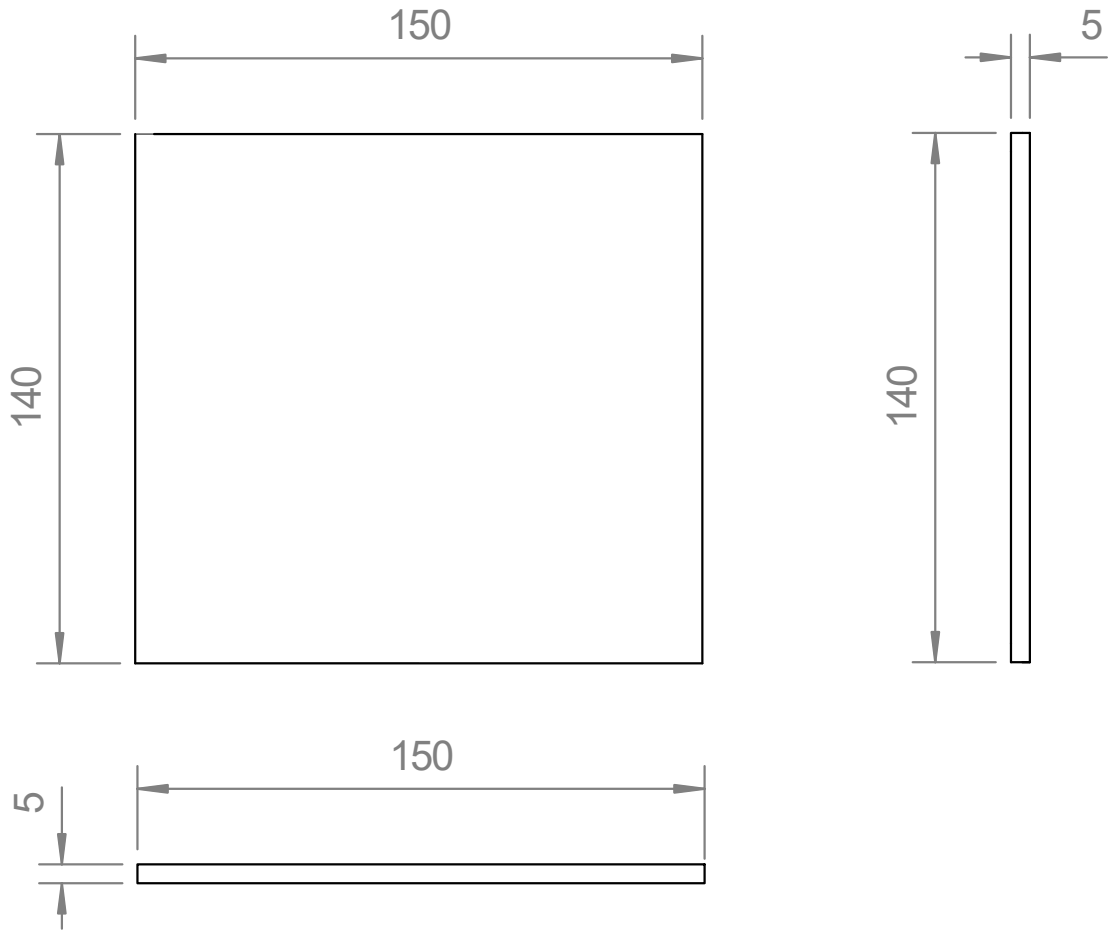
```
#include <Servo.h>
Servo servo1;
Servo servo2;
int maxnum = 110;
int vel = 3;
void setup() {
Serial.begin(9600);
servo1.attach(5);
servo2.attach(7);
servo1.write(0);
servo2.write(0);
}
void loop() {
      for (int i = 0; i <= maxnum; i++)
      {
            servo1.write(i);
            servo2.write(i);
            delay(vel);
      }
      for (int i = maxnum; i >= 0; i--)
      {
            servo1.write(i);
            servo2.write(i);
            delay(vel);
      }
}
```

4. Plànols

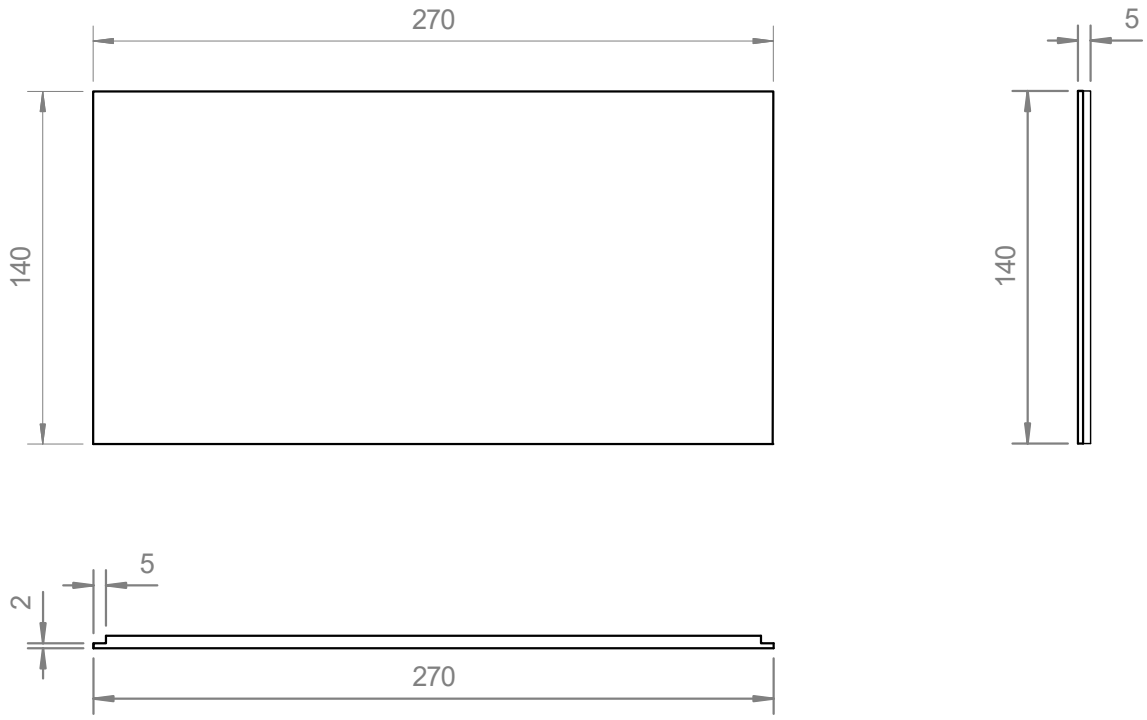
Els plànols són de les peces de fusta que hem hagut de fabricar per tal de construir la caixa. Estan realitzats amb el programa AutoSketch.



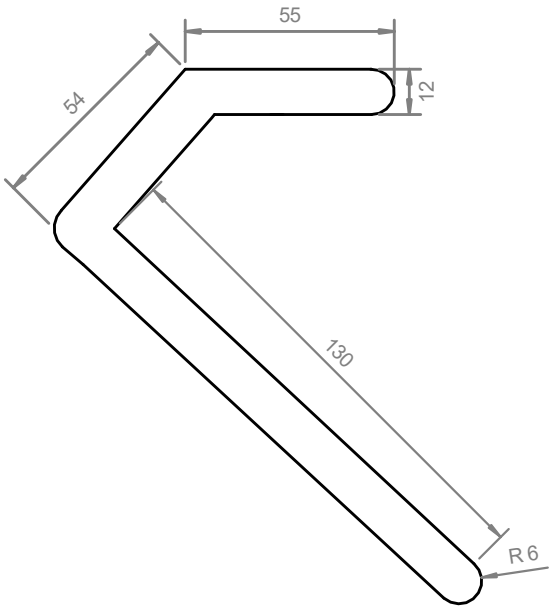
Noms	Marina Anento Ferran Saigí	Curs	2n Batxillerat	INS Les Marines	
		Data	06/02/2018		
Escala 1:4	Peça de la tapa			Treball de recerca	
				Làmina 1	

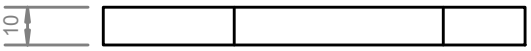


Noms	Marina Anento Ferran Saigí	Curs	2n Batxillerat	INS Les Marines	
		Data	02/06/2018		
Escala 1:2	Peça dels laterals			Treball de recerca	
				Làmina 2	



Noms	Marina Anento Ferran Saigí	Curs	2n Batxillerat	INS Les Marines	
		Data	06/02/2018		
Escala 1:3	Peça dels laterals 2			Treball de recerca Làmina 3	





Noms	Marina Anento Ferran Saigí	Curs	2n Batxillerat	INS Les Marines
		Data	02/06/2018	
Escala 1:2	Peça del dit			Treball de recerca Làmina 4

5. Vídeo

En el següent enllaç està el vídeo "Construint el nostre robot". A l'inici es veuen tots els experiments que vam realitzar durant el curs de robòtica. Després comença el procés de construcció de la *caixa inútil*.

https://youtu.be/yMyLG9Gof-Q