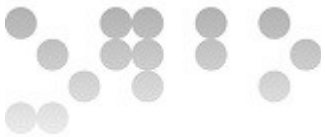
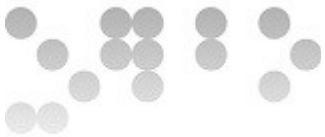
CONTROL DOMÈSTIC VISUAL PER ANDROID

Desenvolupament d'aplicacions de programari lliure

Eduard Calero Rovira
Gregorio Robles Martínez
Oriol Palenzuela i Rosés

10 de Gener de 2015





L'objectiu d'aquest projecte és el desenvolupament d'una aplicació per dispositius mòbils que es comuniqui amb el dispositiu ODControl de l'empresa Opendomo, per tal de controlar un habitatge de manera remota.

A través d'aquesta aplicació l'usuari ha de poder rebre i enviar dades al dispositiu l'ODControl.

El dispositiu ODControl llegeix dades dels diversos sensors i aparells electrònics d'un habitacle a partir de diversos ports i també permet a partir de ports de sortida enviar ordres, com es veurà al llarg del document.

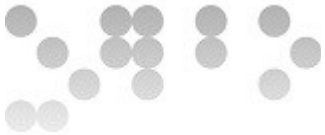
El que es vol doncs, és que a través d'un canal de comunicació via protocol HTTP s'enviïn i rebin dades del dispositiu. La manera de fer-ho ha de ser a través d'una aplicació intuïtiva, el que vol dir implícitament que es controli a través d'un entorn gràfic fàcilment entenedor basant-nos en les aplicacions que podem trobar actualment per a dispositius mòbils.

En obrir l'aplicació, l'usuari ha de trobar una pantalla de *login* on d'introduir credencials, URL o IP del dispositiu i imatge o imatges del plànol.

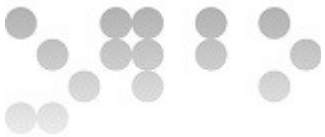
A partir d'aquestes dades el programa ha d'enviar ordres contra l'adreça del dispositiu.

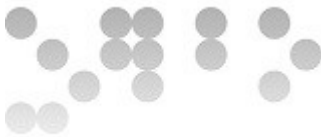
Les ordres a enviar a través d'HTTP són les nadives del propi dispositiu, que també es pot controlar directament connectant-nos a través del port corresponent.

Un cop confirmades les dades introduïdes, el programa ha de permetre a través d'un menú ubicar els diversos sensors i interruptors assignats als ports, sobre el plànol carregat a la pantalla inicial. De manera que el que ha de resultar ha de ser un plànol navegable amb diverses icones, a través de les quals es puguin enviar o rebre dades de manera gràfica i entenedora.



Índex de continguts





1 Introducció

En l'època que vivim actualment, els recursos són cada dia més escassos i la tecnologia és cada cop és mes avançada i present en el nostre dia a dia. Si a això sumem una població mundial creixent, podem deduir que en les properes dècades l'eficiència esdevindrà un element clau en la societat del futur, on el consum de recursos cada vegada haurà d'ésser més i més curós i l'escassetat farà que els preus cada vegada siguin més alts per llei de oferta/demanda dins una economia de mercat.

És en aquest context on la tecnologia ens permetrà consumir-los de manera més eficient ajudant a l'ésser humà a continuar avançant i mirar endavant amb optimisme.

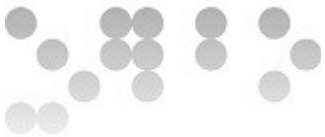
És també la tecnologia la que ens permetrà, de fet ja ho està fent, trobar nous recursos més nets, barats i/o abundants, com és en el cas dels recursos energètics. I és que l'energia és un element importantíssim en la nostra vida i en el nostre dia a dia, es troba present a tot arreu: desde que ens sona el despertador al matí fins que apaguem el darrer llum a la nit per anar a dormir.

Ens hem de preguntar doncs, com podríem aconseguir optimitzar el seu consum? Com ens podria ajudar la tecnologia a aconseguir-ho?

Responent a la primera pregunta de ben segur que a tots se'ns acudiran diverses maneres molt simples d'optimitzar recursos a nivell domèstic: encendre els llums i la calefacció només quan sigui necessari aprofitant al màxim la llum del sol, fer servir electrodomèstics i llums de baix consum o controlar que no hi hagi pèrdues energètiques per un mal funcionament són algunes de les més destacables. Com ens pot facilitar la tecnologia totes aquestes petites accions? La resposta és simple: DOMÒTICA.

1.1 Què és la domòtica

Segons la primera frase a la definició que trobem a la Viquipèdia: «[wk4]*La domòtica consisteix en la creació d'habitatges i edificis automàtics amb la finalitat de millorar-ne la gestió energètica i la qualitat de vida dels seus habitants.*» però jo diria més aviat



que la domòtica no és més que l'aplicació de la tecnologia als habitatges per tal de fer-los més eficients i intel·ligents.

La domòtica ens permet *programar* els edificis, fer-los automàtics o controlar-los remotament amb la fi de fer-los més còmodes i eficients. És per tant una eina que ens permet principalment reduir el consum i consumir de manera responsable.

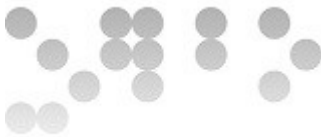
Tot i que ja fa anys que es parla de domòtica, actualment no és un element d'ús generalitzat i popular com pot ser la calefacció o l'aigua corrent als habitatges. Els motius són diversos: l'adaptació dels edificis antics i principalment l'alt cost suposen una barrera perquè la domòtica acabi arribant a tots els habitatges sense distinció com un bé popular.

1.2 L'impacte del software lliure sobre la domòtica

La tecnologia però, ha avançat força tots aquests anys i de fet ho fa exponencialment, sobretot en els darrers 50-70 anys.

Ja han quedat enrere els anys on la tecnologia era una cosa reservada a un reduït nombre de persones i desconeguda pel gran públic. Molt han tingut a veure en això sobretot en el camp informàtic la popularització de les computadores a partir de l'any 1975 amb l'Altair 8800[wk1] i l'aparició de programari sota llicències lliures com els sistemes operatius GNU/Linux o BSD a finals dels anys 80 i principis dels 90. Aquest nou paradigma de la programació que no havia existit fins al moment va permetre estudiar i conèixer el funcionament intern dels programes, donant peu a una popularització i *democratització* de la programació i els ordinadors personals, i no solament en els sistemes operatius sinó també en els estàndards d'internet com ara el protocol TCP/IP.

Aquesta obertura de coneixement a l'exterior ha permès la contribució de persones d'arreu del món gràcies a internet i alhora ha permès també l'assumpció d'aquests coneixements per aplicar-los i millorar-los donant peu a l'aparició de noves tecnologies. No ha quedat exempta la domòtica.



La domòtica tal i com havia passat amb altres tecnologies sota llicències de software privatiu havia quedat inicialment subjecta a les condicions imposades per les empreses i corporacions que la desenvolupaven. És exactament el mateix cas que amb els ordinadors personals. Això volia dir, que el consumidor final estava subordinat totalment i no podia modificar, millorar o corregir el software que s'executava en els dispositius de domòtica. Era totalment presoner de les empreses desenvolupadores.

A més a més, al ser una tecnologia relativament nova, el preu dels dispositius no estava a l'abast de tothom el que en una visió global volia dir que només podrien ser eficients els que disposaven de la capacitat d'invertir en aquests dispositius, cosa que en termes generals perjudica a la societat en conjunt ja que l'energia és un bé escàs i moltes vegades contaminant.

L'aplicació del software lliure, una vegada més, podria ser capaç de salvar tots aquests murs per obrir la tecnologia al món i és en aquest sentit en el que s'inspira el projecte descrit i desenvolupat en aquest document.

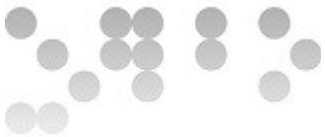
1.3 El projecte

Essencialment el projecte pretén col·laborar en la popularització de la domòtica amb la finalitat de fer-la extensible al major nombre possible de persones per tal de millorar la seva qualitat de vida i convertir-les en subjectes més eficients en termes energètics, fet que es beneficiós per a la societat en conjunt.

La manera de col·laborar serà aplicar els coneixements adquirits durant el màster per tal de millorar un software que controla un dispositiu domòtic desenvolupat sota filosofia de software lliure o codi obert, per una empresa del nostre país anomenada Opendomo.

1.4 Estructura

El document està estructurat en 6 capítols. El primer i actual, on s'introdueix i es contextualitza el projecte i 5 capítols més. Al llarg del segon capítol es descriuen els



objectius del projecte mentre que al tercer veurem l'estat de l'art de la domòtica. Dins d'aquest capítol trobarem un breu resum de la situació actual i les empreses que la desenvolupen així com de les tecnologies emprades per al desenvolupament del nostre projecte, essencialment llenguatges de programació i protocols informàtics.

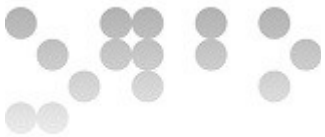
Al quart capítol trobarem el gruix del projecte i és on queda descrit el disseny i desenvolupament del software, dins d'aquest capítol hi ha diverses seccions: una primera que conté una breu introducció del dispositiu per al que desenvoluparem el software, una segona que comenta la preparació de l'entorn de treball per tal de poder començar a programar i fer proves i després successives seccions i subseccions que comenten detalladament el software desenvolupat. Seguidament trobarem el cinquè capítol on es comenten els resultats obtinguts, on es comenta com funciona el programari a nivell usuari. Finalment al darrer capítol trobem les conclusions generals del projecte.

Addicionalment trobem la bibliografia sobre la que es referencien les dades extretes de documents i fonts alienes al present document a través de superíndexs.

1.5 Referències bibliogràfiques

Al final del document trobem la bibliografia. Totes les referencies i fonts utilitzades i referenciades al llarg d'aquest document consten en aquell apartat referenciades a través d'un codi de 3 dígits alfanumèrics en forma de superíndex.

Cada vegada que trobem en el text un d'aquests superíndex, només cal anar a la bibliografia per saber a quin element bibliogràfic fa referència a través del codi.



2 Objectius

Es vol desenvolupar una aplicació amb una interfície gràfica per a dispositius mòbils a través de la qual l'usuari es pugui comunicar amb el mòdul ODControl .

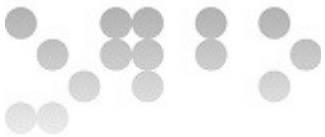
Tot i que es podria fer programant en llenguatge nadiu de cada dispositiu, que en el cas que ens ocupa seria llenguatge nadiu Android (java), com veurem més endavant ens servirem d'una nova tecnologia anomenada Phonegap que ens permetrà programar en *HTML5*, *CSS3* i *JavaScript* i adaptar-ho a la majoria de sistemes per a dispositius mòbils existents.

L'aplicació ha de satisfer els següents requeriments:

Primerament cal tenir present que a l'hora de carregar el plànol de l'habitacle a controlar, aquest pot ser en qualsevol resolució i per tant en funció del *display* del dispositiu, aquest es pot veure més gran o més petit fet que pot significar una dificultat a l'hora de llegir sensors, interactuar amb ells o entendre el plànol ja que pot quedar gegant o diminut. A causa d'això haurem de trobar una solució per poder moure'ns dinàmicament sobre el plànol i poder ampliar-lo o reduir-lo. El que es vol és obtenir una navegació fluida similar a la que trobem actualment en els navegadors mòbils, google maps o el carrets fotogràfics dels dispositius mòbils d'última generació.

En segon terme l'aparell pot controlar i monitoritzar elements situats en diverses plantes d'un mateix edifici, per tant caldrà trobar una solució també a aquesta possible situació, permetent carregar i mostrar diverses plantes d'un mateix edifici, per exemple fent possible introduir diversos plànols per diverses plantes i poder moure'ns entr ells.

Un altre element a tenir en compte és l'espera. Perquè una aplicació tingui èxit, la seva fluïdesa a l'hora d'interactuar té molt a veure. Molt sovint si l'usuari detecta que ha d'esperar massa, pot pensar que el funcionament de l'aplicació és deficient o anormal. En el cas d'aplicacions que es comuniquen amb elements exteriors, la latència en la resposta queda fora de l'abast del desenvolupador, ja que depèn molt de la connectivitat de l'aparell. Tot i això el que si que es pot minimitzar és l'espera mostrant



per pantalla elements en moviment mentre s'intercanvien dades. De manera que l'usuari no té la sensació de que l'aplicació no funciona correctament sinó que perceb que s'està comunicant o s'estan carregant dades. Tenint en compte que aquesta aplicació es comunica amb un aparell remot ho haurem de tenir en compte per implementar algun mecanisme d'aquest tipus.

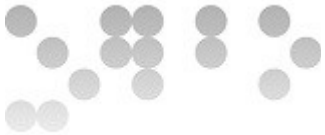
Pel que fa als objectius que tenen relació amb les competències i aptituds a nivell personal i professional, s'espera desenvolupar la capacitat per afrontar la resolució de problemes i projectes tecnològics a partir dels coneixements adquirits al llarg del màster, principalment amb la filosofia del software lliure i de codi obert.

També ampliar l'experiència i els coneixements utilitzats per desenvolupar el projecte sobretot en la vessant de programació i creació d'aplicacions: saber com és tot el procés desde que es pensa el que es vol aconseguir fins que s'aconsegueix el resultat final i es publica, saber dissenyar correctament la part *frontend i backend*, controlar la organització i distribució dels temps del procés i aconseguir un producte final funcional i professional.

Finalment el darrer objectiu és acumular experiència per aprendre a ser totalment autosuficient en la resolució dels problemes que es vagin presentant al llarg del procés sabent aprofitar tots els recursos disponibles de manera autodidacta: fòrums, comunitats, pàgines web, manuals, etc.

2.1 Esquema d'objectius

1. Objectius a nivell funcional: Desenvolupament d'una aplicació per a dispositius mòbils per al control remot del dispositiu ODControl.
    a) Intuïtiva
    b) Funció zoom i navegació sobre els plànols
    c) Diverses plantes
    d) Pantalla de *loading*
2. Objectius a nivell personal
    a) Resolució de problemes aplicant coneixements adquirits



b) Adquisició d'experiència

c) Auto-aprenentatge

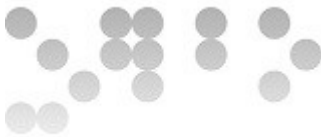
2.2 Planificació

Per tal d'assolir els nostres objectius, desglossarem tot el projecte en diverses tasques i subtasques que queden reflectides en el següent quadre.

Aquest és el quadre inicial on queden reflectides les tasques de manera genèrica i que anirà variant: afegint, eliminant i modificant tasques i subtasques a mida es vagin plantejant nous problemes i replantejaments.

Al capítol 6 dedicat a les conclusions podem trobar un diagrama molt més detallat i ajustat a les tasques que finalment s'han anat desenvolupant fruit de les diverses modificacions.

Tasca	Inici	Fi	Duració	Observacions
Fase 1: Desenvolupament (HTML5+CSS3+Javascript)	**03/30/14**	**07/04/14**	**84**	
Revisió de coneixements teòrics	03/30/14	05/03/14	31	Durant aquesta primera etapa de la fase 1 procedirem a aglutinar i posar al dia els coneixements necesaris.
Revisió del funcionament del dispositiu	05/05/14	05/10/14	6	Adquisició de coneixements
Preparació de l'entorn de desenvolupament	05/13/14	05/17/14	5	Configuració del nostre IDE i familiarització amb l'entorn.
Proves de funcionament	05/19/14	05/24/14	6	
Desenvolupament inicial en format web	05/26/14	06/14/14	18	Primer disseny en format web.
Modificacions i adaptacions	06/16/14	06/21/14	6	Primeres modificacions seguint el criteri del tutor.
Memòria (part 1)	06/23/14	07/04/14	11	Redacció de la primera part de la memòria.
Fase 2: Phonegap	**09/14/14**	**11/20/14**	**59**	
Adquisició de coneixements teòrics	09/14/14	09/19/14	6	Adquisició de coneixement del funcionament d'aquesta tecnología.
Adaptació de l'entorn de desenvolupament	09/20/14	09/26/14	6	Adaptació del nostre entorn de desenvolupament a phonegap.
Proves de funcionament	09/27/14	10/03/14	6	Primeres proves de funcionament amb phonegap.
Adaptació	10/04/14	10/28/14	21	Adaptació de la versió inical a la versió mòbil inicial.
Modificacions	10/29/14	11/10/14	11	Modificacions seguint el criteri del tutor.
Memòria (part 2)	11/11/14	11/20/14	9	Redacció de memòria
Fase 3: Connexió	**11/23/14**	**01/10/15**	**43**	
Proves de funcionament	11/23/14	11/28/14	6	Proves finals de funcionament i usabilitat
Modificacions finals	11/29/14	12/17/14	16	Rectificació d'errades i modificacions finals
Memòria (Part 3)	12/18/14	01/10/15	21	Redacció de la part final de la memòria



3 Estat de l'art

Durant l'elaboració d'aquest document s'ha anat recopilant informació en relació a l'estat actual en l'àmbit de la domòtica i les tecnologies i recursos emprats per la consecució del projecte. En aquest apartat intentarem sintetitzar l'estat actual en aquests camps de manera breu i entenedora.

3.1 Domòtica

El terme domòtica és un terme molt ampli que engloba diversos elements emprats amb la finalitat d'aconseguir edificis intel·ligents, automàtics i/o eficients. Hauríem de parlar doncs del terme *sistema domòtic* i dels elements que el componen.

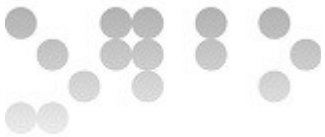
Podríem dir que un sistema domòtic bàsic consta de 3 elements: *gateway*, mitjà de comunicació i dispositius de control i monitoratge.

- El **gateway** és l'element a través del qual ens comuniquem amb el sistema per rebre i enviar dades ja siguin de control o lectura.
- **El mitjà de comunicació** és el medi a través del qual circulen les dades i el seu format (protocol).
- **Els dispositius** són els elements que ens permeten interactuar amb l'habitacle ja sigui llegint dades d'algun sensor de tipus lumínic, de moviment, temperatura, etc. o enviant ordres com obrir una persiana o encendre un llum.

Trobem multitud de empreses relacionades amb a la domòtica:

D'una banda un tipus d'empreses que ho fa desde un punt de vista de disseny i implementació del sistema, és a dir, s'encarreguen de tot el procés d'implantació desde que el client indica quines són les seves necessitats, fins que s'instal·la el sistema, passant per la fase de disseny del sistema i adquisició i selecció dels equips necessaris. D'aquestes se'n poden trobar un munt sense haver de buscar gaire a internet.

D'altra banda trobem les que es dediquen a la comercialització de dispositius de tipus



domòtic, és a dir, fabriquen i comercialitzen cadascun dels dispositius que conformen el sistema domòtic.

Per formar el sistema domòtic, generalment aquests dispositius necessiten comunicar-se i entendre's entre ells però actualment no hi ha un estàndard de comunicació *wireless* per aquest tipus de dispositius sino que hi ha diferents proposats desde diferents empreses o grups d'empreses que competeixen per imposar el seu. Z-Wave, RFXCOM, RTS o EnOcean són alguns d'ells.

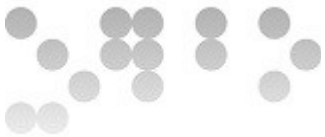
Queda palès si investiguem una mica que el paradigma més estès a l'hora de dissenyar, fabricar i comercialitzar aquests dispositius és dissenyar un aparell, sota una llicència privativa sobre el que s'executa software privatiu, fet que és lògic ja que les empreses que els desenvolupen i comercialitzen esperen tenir un retorn econòmic en la seva inversió de desenvolupament de software i hardware en el moment de la venda, però hi ha altres models basats en *hardware lliure* i/o software lliure.

Que volem dir amb *hardware lliure*? El hardware lliure es un hardware amb un disseny llicenciat sota llicència lliure o de codi obert. A diferència del que succeeix amb el software la implementació del disseny suposa la creació d'un objecte físic i per tant hi ha algunes lleis de la filosofia lliure o de codi obert que s'interpreten de manera diferent a la d'un software, com per exemple la de lliure modificació i distribució, ja que la implementació de canvis o creació física del mateix hardware suposa un cost implícit en els seus components.

Dins del hardware lliure cal destacar dos dissenys molt populars dins el món de la domòtica.

3.1.1 Raspberry pi

Segons la wikipedia «[wk2]*El Raspberry Pi és un ordinador monoplaca de baix cost desenvolupat en el Regne Unit per la Fundació Raspberry Pi. L'objectiu principal d'aquest disseny és estimular l'ensenyança de les ciències de la computació, però*



també s'ha popularitzat com a plataforma per a dissenys d'aficionats i per a usos informàtics generals. Els primers models es van començar a comercialitzar el febrer de 2012. La darrera versió, Raspberry B+, s'ha anunciat el juliol de 2014 tot mantenint el mateix processador però amb diverses millores en els seus connectors i un menor consum energètic.[...]»

Veurem que hi ha aparells comercials que permeten adaptar aquest ordinador monoplaca per al control domòtic de manera que es pugui comunicar i enviar ordres a la resta de dispositius fent la funció de *gateway,* com per exemple el *RaZberry500* (*http://www.z-wave.com/find_products/gateways*) fet servir per alguns productes de l'empresa francesa Jeedom.

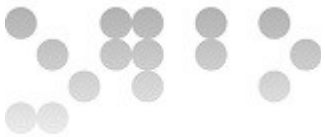
3.1.2 Arduino

A diferència de la Raspberry Pi, Arduino és només un microcontrolador i per tant les seves opcions són més limitades que en el cas de Raspberry. De fet, hi ha diferents versions de placa que consten generalment d'un microcontrolador AmtelAVR i ports d'entrada i sortida.

Aquesta placa és també molt popular dins el món de la domòtica, de fet el sistema DOMINO que corre sobre els ODControl de Opendomo, es pot carregar fàcilment sobre aquesta placa i fer-lo servir de manera similar com passaria amb aquest dispositiu. (http://es.opendomo.org/domino)

3.2 Model lliure a la domòtica

Tal i com comentàvem anteriorment, el model més estès dins del les empreses de domòtica és el model basat en hardware i software privatiu. Aquest model però com ja s'ha es comenta a l'apartat 1.2 d'aquest mateix document, deixa subjecte a l'usuari final a les restriccions del desenvolupador. Que vol dir això? Posarem alguns exemples per tal de que s'entengui d'una manera més pràctica:



Suposem que l'usuari compra un dispositiu tipus *gateway. A* través d'aquest dispositiu pretén controlar una sèrie de dispositius que té instal·lats al seu habitatge però resulta que el software tot i disposar de un sistema *wireless* per comunicar-se sense fils, fa servir el seu protocol de comunicació privatiu. Això implica que l'usuari ja estarà subjecte a fer servir dispositius compatibles amb aquest protocol, que normalment serà compatible amb productes de la mateixa empresa, si vol tenir el sistema controlat d'una manera centralitzada. **Primera restricció: Incompatibilitat**.

Suposem que tot i ser privatiu, el protocol de comunicació esta bastant acceptat com a estàndard de diverses empreses com pot ser el cas de Z-Wave, però l'usuari vol controlar un nou aparell que ha comprat i que fa servir exactament el mateix protocol. Desgraciadament la empresa que va desenvolupar el dispositiu encara no ha implementat cap actualització del firmware de l'aparell per que es pugui controlar. L'usuari haurà d'esperar a que l'empresa ho faci, en cas de que estigui disposada.

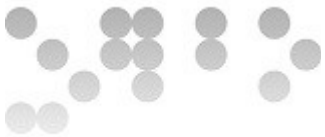
Segona restricció: Dependència.

Per donar ordres al dispositiu l'usuari es connecta al cloud de l'empresa desenvolupadora on a través de una interfície web dona ordres, automatitza processos, llegeix dades, etc. Això vol dir que les dades de tot el que succeeix al seu domicili surten de la LAN o WAN de l'habitatge i viatgen a traves d'internet fins a l'empresa fabricant del dispositiu, sense saber que es fa amb elles i depenent totalment de que l'empresa tingui un sistema que garanteixi la confidencialitat de les dades, exposant l'usuari a un risc totalment innecessari.

Tercera restricció: Garantia total de confidencialitat de les dades.

Segurament n'hi ha d'altres. Però només aquestes 3 ja donen força sentit a la implantació del model de filosofia lliure dins la domòtica.

Tot i això encara són poques les empreses i projectes dedicats a aquest model. Opendomo Services és una d'elles i és en el context d'aquesta empresa on es



desenvolupa el present projecte descrit en aquest document. Un altre exemple que podríem donar és la francesa Jeedom (https://jeedom.fr/index.php), nascuda a finals del 2013.

3.2.1 Opendomo Services SL

Actualment ja hi ha empreses que es dediquen plenament a la domòtica basant-se en la filosofia del software lliure o de codi obert. Aquesta filosofia permet oferir dispositius amb un software totalment lliure, és a dir, utilitzable com es vulgui, modificable i compartible, a un preu molt raonable.

Una d'aquestes empreses és Opendomo.

Opendomo dissenya, desenvolupa i ven dispositius per al control i monitoratge d'instal·lacions elèctriques (llar i empresa). A més d'oferir tot tipus de serveis en relació als seus productes (assessorament, adaptació, desenvolupament, I+D, etc.).

Tots els seus productes estan creats en base a la filosofia de hardware i software lliure o de codi obert. En base a aquests elements l'empresa aplica els seus coneixements tecnològics per adaptar-los a les necessitats dels clients i/o desenvolupar-los amb l'objectiu d'orientar-los a cobrir noves necessitats.

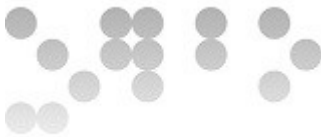
Òbviament l'empresa aprofita els beneficis de treballar sota llicència lliure com ara beneficiar-se de les contribucions de la comunitat per créixer més ràpidament.

Podríem agrupar els productes d'Opendomo en 3 grups:

- **Control de consum energètic (Energy):**

Aquesta gamma de productes està constituïda per diverses versions del mòdul ODEnergy a través del qual el sistema elèctric, ja bé de la casa o l'empresa o l'espai que es vulgui controlar, es comunica amb l'usuari.

De manera que, d'una banda el dispositiu comunica amb el sistema elèctric i de l'altra a la xarxa (LAN/WAN), llegint el consum elèctric, interpretant-ne els resultats a través d'algoritmes i enviant les dades processades a l'usuari per controlar el consum i ser més eficients energèticament.



- **Control remot de dispositius (Control):**

Dins aquest grup inclouríem únicament un producte: ODControl. Aquest dispositiu ens permet interactuar amb el sistema elèctric de manera remota enviant i rebent senyals digitals. De manera que es poden remotar accions com ara obrir i tancar llums, pujar i baixar persianes, control de la calefacció i de qualsevol dispositiu que es pugui comunicar amb el mòdul i disposi d'un accionament electrònic.

- **Monitoratge de sensors (Gateways):**

Dins aquest grup inclouríem únicament un producte: OD485.

Aquest dispositiu permet llegir multitud de mesuradors i sensors i interpretar-ne els resultats per tal de fer-los arribar a l'usuari: sensors de moviment, temperatura, etc.

3.2.2 Jeedom

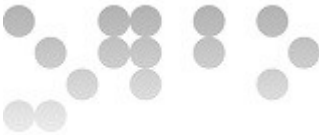
Jeedom és una projecte francés de domòtica lliure iniciat a finals del 2013 per Loic (loic@jeedom.com) i que es va llançar oficialment el 24 de juny de 2014.

Actualment l'equip esta fomat per un membre més: Mathieu (mathieu@jeedom.com).

Sembla ser que inicialment el projecte tenia com a objectiu desenvolupar un software per programar plaques basades en hardware lliures com Raspberry i que fessin funcions d'automatització i control domòtic, però actualment també es dediquen al desenvolupament de dispositius.

Els principals punts forts dels sistemes basats en aquest software estan estretament relacionats amb els avantatges que suposa la incorporació de software lliure:

- **És open source** amb tot el que això suposa: estudi, modificació, millora, etc.
- **És multi-protocol**: suporta Zwave, RFXCOM, RTS SOMFY, En Ocean, xPL i d'altres.
- **Garantía de confidencialitat.** Gràcies a que la gestió es fa a nivell local i no a través de servidors externs.



- **Jeedom Marquet** que disposa de interficies personalitzables i nous mòduls.
- **Personalitzable**. Aquest punt es deriva del primer.

3.3 Tecnologies web

Un cop situats en el context tecnològic i en l'entorn en el que situarem el projecte. Caldrà saber també quina és la situació de les principals tecnologies que farem servir per tal de dur-lo a terme.

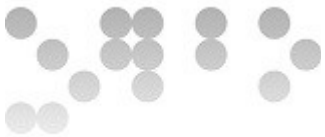
3.3.1 Phonegap

Tot i que el que es pretén desenvolupar al llarg del projecte és, al cap i a la fi, una aplicació per a dispositius mòbils, no farem servir els llenguatges nadius de desenvolupament d'aplicacions de cadascun dels sistemes operatius d'aquests dispositius sinó que ens servirem d'una tecnologia que ens permetrà desenvolupar una sola aplicació fent servir tecnologies web estàndard (HTML+CSS+Javascript) i adaptar-la a tots els sistemes. Aquesta tecnologia és Phonegap.

Phonegap és un Framework de desenvolupament web (mòbil) basat en el projecte lliure Apache Cordova, que inicialment va ser desenvolupat per l'empresa americana Nitobi i posteriorment comprat per Adobe (http://phonegap.com/2011/10/03/nitobi-enters-into-acquisition-agreement-with-adobe-2/).

Phonegap proporciona una API estàndard multiplataforma que es la que es comunica amb els sensors dels dispositius de manera que fa de pont entre la petició des de el codi web i el dispositiu, proporcionant així la possibilitat de fer servir el mateix codi per a diverses plataformes.

De fet el que fa un aplicació escrita en Phonegap no és més que un cop executada, dispara el navegador predeterminat del SO (WebView) en un envoltori d'aplicació nativa i comença a interpretar el codi web. En el moment en que li cal comunicar-se amb els sensors del telèfon (bidireccionalment) fa servir la API que implementa les funcions a llenguatge nadiu a través de `Phonegap.js` (o `Cordova.js`).



A nivell de codi, tot i que la flexibilitat és total (atès que disposem de tota la potència que ens proporcionen les tecnologies abans comentades), actualment ja s'imposen una sèrie de tècniques que satisfan les habituals necessitats que presenta una aplicació mòbil.

Són aquestes tècniques les que ens permetran resoldre alguns dels principals reptes que presenta el desenvolupament.

3.3.2 HTML5

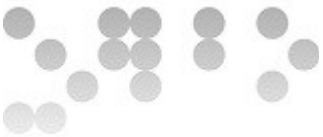
«[wk3] *HTML (acrònim d'Hyper Text Markup Language, en català, "llenguatge de marcat d'hipertext"), és un llenguatge de marcat que deriva de l'SGML dissenyat per estructurar textos i relacionar-los en forma d'hipertext. Gràcies a Internet i als navegadors web, s'ha convertit en un dels formats més populars que existeixen per a la construcció de documents per a la web.*»

A través d'aquestes marques o etiquetes s'estructura el text de manera que el navegador pot donar-li un format i relacionar textos entre ells. Podríem dir que és el format per excel·lència a internet.

Tot i que HTML fa referència només a la part de llenguatge de marcat, que és la part que estructura el text, aquest es combina amb les tecnologies *CSS* i *Javascript* (comentades als següents apartats) per poder oferir estructura, estil i funcionalitat com una eina integrada. És a dir, aquestes tecnologies funcionen com un sistema, ja que estan totalment relacionades i és per això que ja en la 5a versió, es fa servir moltes vegades el nom d'HTML5 per referir-se a tot el sistema o a la combinació de les tres tecnologies esmentades en conjunt.

Pel que fa a les principals millores d'aquesta versió respecte de versions anteriors són:

- Noves etiquetes per tal d'identificar i donar format de mode més eficaç al model de caixa tradicional: <head>, <nav>, <section>, <aside>, <footer>, etc.
- Nous elements per a la reproducció i presentació d'elements audiovisuals: <canvas>, <video>, <audio>, etc.



- Millores en els formularis amb l'aparició de nous tipus d'inputs i atributs: email, search, url, number, tel, range, datetime, placeholdre, required, multiple, autofocus, pattern, form, etc.
- Noves APIs: Canvas, Drag&Drop, Geolocation, Web Storage, IndexDB, File, etc.

3.3.3 CSS3

CSS és l'acrònim de Cascade Style Sheets o el que seria el mateix, fulls d'estil en cascada. Aquesta tecnologia és la que permet atorgar estil al text, és a dir, la part que s'encarrega de la presentació: el disseny.

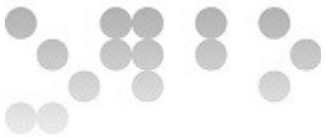
Tot i que en les primeres versions de l'especificació d'HTML hi havia atributs que permetien proveir d'estil als textos, a mida que el model ha anat avançant i les exigències de disseny han anat creixent, s'han anat separant de manera més modular deixant la part de disseny a CSS i aconseguint així que seguint aquest paradigma s'hagi augmentat la potència de CSS permetent per exemple modificar de manera molt més eficaç el disseny d'un text o podent exportar un disseny existent a d'altres textos, entre altres virtuds .

Actualment ja en la seva 3a versió les principals millores respecte a les anteriors versions són:

- Noves propietats de presentació: cantonades arrodonides, ombres en caixes i text, inserció de fonts específiques de text, gradient linial i radial, etc.
- Propietats dinàmiques: escalar, rotar, traslladar, modificar la simetria, controlar la velocitat de les transicions, etc.

3.3.4 Javascript

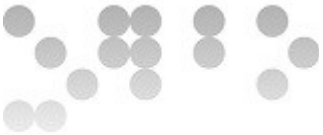
Quan parlem de desenvolupament de pàgines web i HTML5, Javascript és el llenguatge de programació que interpreta el navegador i que permet junt amb



l'estructura proporcionada per HTML i el disseny implementat per CSS programar accions que interactuïn per si mateixes o amb l'usuari creant dinamismes i funcionalitats a la pàgina.

Pel que fa a les millores de Javascript són principalment noves formulacions en la referenciació dels elements HTML i la generació d'events.

També en aquest camp caldria incloure totes les noves APIs de HTML5 que també hem comentat en el apartat HTML (Canvas, Drag&Drop, etc.) atès que HTML implementa només l'objecte mitjançant l'etiquetatge i és a través de Javascript que s'implementa i programa tota la part funcional, el que seria el motor.



4 Disseny i Implementació

Al llarg del present capítol es resumeix tot el procés de disseny, desenvolupament i implementació del software.

Per tal d'entendre com funciona el dispositiu i amb quines eines es treballarà hi ha dos capítols introductoris que resumeixen breument els aspectes importants a tenir en compte del dispositiu i l'entorn de desenvolupament.

A partir d'aquí la resta de capítols descriuen la part de disseny i desenvolupament.

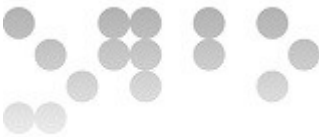
Donat que en el desenvolupament d'un software d'aquest tipus hi ha constants modificacions que de vegades obliguen a replantejar parts ja implementades, s'ha intentat seguir un ordre cronològic que faci el text el més entenedor possible i de vegades s'ha omès codi d'algunes petites modificacions o correccions en parts ja explicades anteriorment i que no tenen rellevància important a l'hora d'entendre com s'ha desenvolupat i com funciona l'aplicació. Aquests detalls dificultarien la correcta estructuració del document i el seu enteniment. Ens referim a aspectes com poden ser modificació de noms de variables i funcions, algunes petites modificacions en el disseny, ordre diferent en la implementació de funcions, etc.

Totes aquestes modificacions i correccions ja queden reflectides en l'aplicació a nivell funcional i es poden llegir dirèctament al codi.

4.1 El dispositiu ODControl

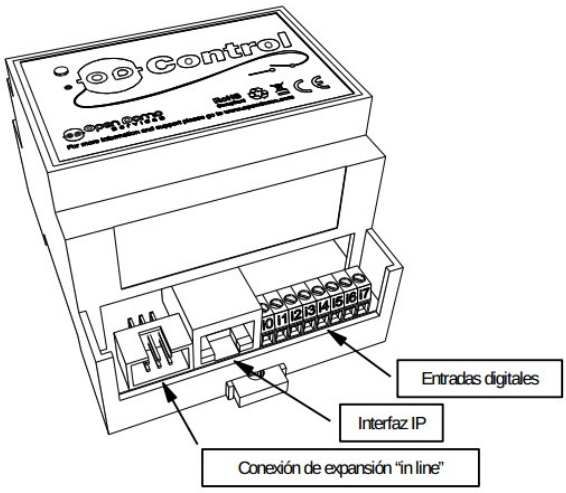
ODControl és una de la família de mòduls desenvolupats per Opendomo. Aquesta família de mòduls constitueix la branca de control, és a dir, es tracta d'uns mòduls configurables que permeten emetre i rebre senyals a través d' una xarxa IP i alhora comunicar-se amb diversos dispositius d'una manera bidireccional, o el que és el mateix, rebent i enviant senyals de/cap als dispositius.

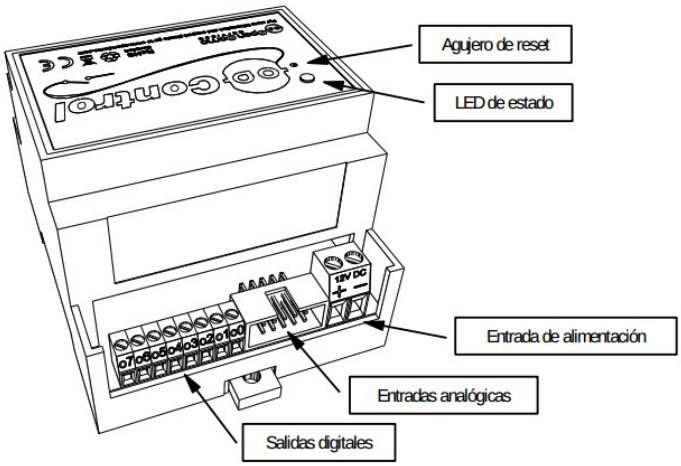
ODControl consta de:



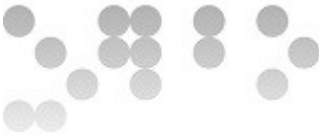
- Ports externs: ports d'entrada/sortida que poden ser analògics i/o digitals.
- Ports virtuals: són ports intermedis que permeten fer diverses accions.
- Enllaços: Són mecanismes que permeten transferir valors entre ports de diverses maneres en funció de les necessitats.

4.1.1 Característiques del dispositiu





Tal i com mostra la figura anterior el dispositiu físicament consta de:

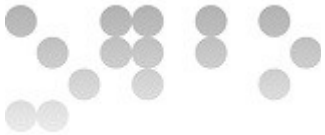


- Entrada d'alimentació a través de cables positiu i negatiu.
- 1 Botó físic de reset.
- 1 LED bicolor d'estat.
- 1 Port RJ-45 10/100M a través del qual ens hi comunicarem per configurar-lo.
- 2 Ports d'expansió (1 intern + 1 extern)
- 8 ports digitals d'entrada
- 8 ports digitals de sortida
- 8 ADC (Entrada analògica que es convertida a digital)

El dispositiu disposa de 2 clips de muntatge a la part superior i inferior per muntar-ho a carril DIN de 35mm, en cas contrari disposa de forats per muntar-lo amb cargols.

A continuació es mostra un quadre amb un resum del rang de valors admesos pel dispositiu extret del *datasheet* del dispositiu:

Paràmetre	Mínim	Típic	Màxim
Voltatge d'alimentació	8Vcc	12Vcc	14Vcc
Corrent d'alimentació	150mA	180mA	1,5A
Voltatge ports digitals de sortida	Vcc-0,7		12Vcc
Corrent ports digitals de sortida			170mA
Potencia dels ports digitals de sortida			2W
Voltatge dels ports digitals d'entrada	0V		30V
Valor "1" dels ports digitals d'entrada	5V	12V	30V
Valor "0" dels ports digitals d'entrada	0V	0V	1,5V
Voltatge dels ports ADC	0V		+2,5V



4.1.2 Modes de funcionament

Control Manual

Aquest és el mode de funcionament en que les ordres als dispositius i la lectura dels sensors es fa a través d'un dispositiu connectat a la xarxa que accedeix al dispositiu via IP, com per exemple un smartphone.

Funcionament autònom

Aquest és el mode de funcionament en que el dispositiu a partir d'una configuració prèvia reb les dades dels diversos dispositius, les processa i envia unes ordres com a resultat de les lectures obtingudes en base a la configuració prèviament programada.

Integració amb Opendomo

Aquest és el mode de funcionament en que ODControl s'integra dins una xarxa ODNetwork permetent el control unificat a través de d'interfície principal.

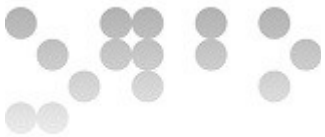
4.1.3 Comunicació i comandes

El dispositiu es comunica amb l'usuari a través de la xarxa IP i és accessible a través de **port 80 (http)**. L'adreça per defecte és http://169.254.0.1.

L'**usuari - password** per defecte és: **user – opendomo**.

A través del **port 80** es pot configurar el dispositiu a través de comandes. Aquest tipus de comunicació es transmet **sense xifratge** i per tant es recomana accedir de manera local. Per executar la comanda *versió (ver),* per exemple, faríem:

```
http://169.254.0.1:80/ver+
```



> És important recordar que s'han de substituir els espais en blanc per el signe '**+**'

A continuació es comenten breument les comades més destacades.

4.1.3.1 Versió (`ver`)

`http://169.254.0.1:80/ver+`

Mostra la versió de del *firmware* que hi ha carregat al dispositiu, junt amb la seva data de creació i el nom del dispositiu.

4.1.3.2 Llistat (`lsc`)

`http://169.254.0.1:80/lsc+`

Aquesta comanda ens fa un llistat complert de tots els ports. El format retornat és el següent:

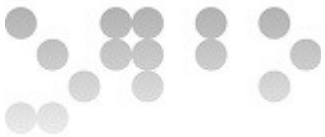
Cada port consta de 3 camps. Cada camp està separat per el signe '**:**' (dos punts).

El primer camp està format pel nom del port. Aquest camp es pot modificar, però per defecte consta de dues lletres i 3 xifres. Les lletres indiquen el tipus de port i les xifres serveixen per distingir entre ports del mateix tipus.

El segon camp consta de 4 caràcters:

Els dos primers indiquen el tipus de port segons la següent taula:

Caràcters	Tipus	Modificable
DO	Sortida digital	-
DI	Entrada digital	-
AO	Sortida analògica	-
AI	Entrada analògica	-
DV	Virtual digital	Si



AV	Virtual analògic	Si
Dv	Virtual digital	No
Av	Virtual analògic	No
TV	Virtual temporitzador	-
WV	Virtual programable per hora i dia	-
CV	Virtual programable per data	-
Rv	Virtual dispositiu remot	No

El tercer caràcter pot tenir 3 valors:

- **M**: Visible a la pestanya *Home.*
- **C**: Visible a la pestanya *Config*.
- **H**: Ocult.

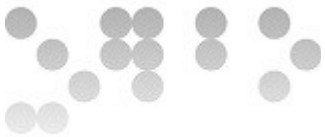
Finalment l'últim caràcter indica el *tag* en cas que s'hagi configurat o en el seu defecte mostra un signe '_' (barra baixa).

Pel que fa al tercer camp correspon al valor del camp, que en ports digitals serà *ON* o *OFF* i en camps analògics un valor en nombres.

En el cas dels ports analògics encarà hi hauran dos camps més:

- Un primer camp que indicarà el rang de valors, és a dir, valor màxim i valor mínim, separat pel caràcter '|' (barra vertical).
- Un segon i darrer camp que ens indicarà la seva tolerància.

En ports virtuals, el quart camp indica el tipus de port virtual.



4.1.3.3 Establir (`set`)

Aquesta comanda serveix per assignar un valor a un port, per tant només es podrà fer servir sobre ports de sortida (o virtuals). Els seu ús és molt simple:

`set` *`nomport valor`*

`http://169.254.0.1:80/``set+`*`nomport+valor`*

`Valor` en ports digitals podrà tenir els valors `on/off` i en ports analògics s'introduirà el valor separant els decimals amb un punt o bé indicant el seu valor en deumil·lèsimes.

El rang de valors acceptats va del `99999999` (en decimals `+9999.9999`) al `-99999999` (en decimals `-9999.9999`).

4.1.3.4 Reload (`rel`)

`http://169.254.0.1:80/rel+`

Aquesta comanda reinicia el dispositiu, tal i com succeeix al mantenir apretat el botó físic de *reload* de l'aparell.

4.1.3.5 Clear (`clr`)

`http://169.254.0.1:80/clr+`

Aquesta comanda esborra les configuracions de ports i enllaços, conservant la IP, el nom del dispositiu i les credencials de usuari.

4.1.3.6 Default (`def`)

`http://169.254.0.1:80/def+`

Aquesta comanda aplica la configuració per defecte.

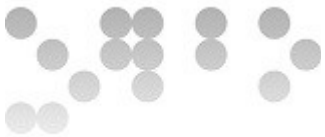
4.2 Preparació de Phonegap

Tenint en compte la tecnologia que farem servir (HTML5+CSS+Javascript), cal tenir present que podríem treballar directament sobre un arxiu de text, compilar-lo i obtindríem el binari del programa a través de cadascun dels elements compilats de cada plataforma. Fent servir Phonegap, com veurem a continuació, fins i tot obtindríem totes les versions binaries disponibles, per cadascun dels SO mòbils que actualment trobem al mercat, o gairebé.

4.2.1 Instal·lació de Phonegap/Cordova

A continuació descriurem breument els passos a seguir per instal·lar Phonegap en Ubuntu:

Per instal·lar Phonegap primerament cal instal·lar Node.js:



En el cas d'Ubuntu, abans de compilar el programa cal assegurar-se de tenir instal·lats els prerequisits:

- `GCC >= 4.2`
- `Pyton 2.6 o 2.7`
- `GNU Make`

A continuació es compila:

```
./configure
make all
sudo make install
```

Un cop instal·lat `node.js` ja es pot fer servir l'ordre `npm` per instalar Phonegap de manera global:

```
sudo npm install -g phonegap
```

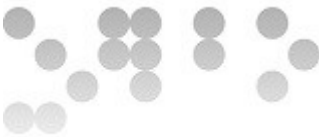
Un cop instal·lat Phonegap i abans de treballar cal fer dues darreres passes abans de començar a compilar amb ell:

- Comprovar que tenim `java` la llibreria `ant` instal·lada:

  ```
  sudo apt-get install openjdk-7-jdk ant
  ```

- Pasar al sistema les variables d'entorn perquè trobi correctament les `SDK tools i les Platform-tools` quan compilem Àndroid en local desde Phonegap. Dins de la carpeta SDK de l'ADT d'Àndroid.

  ```
  export ANDROID_HOME="$HOME/Escritorio/adt-bundle-linux-x86-20131030/sdk/tools"
  export ANDROID_PLATFORM_TOOLS="$HOME/Escritorio/adt-bundle-linux-x86-20131030/sdk/platform-tools"
  export PATH="$ANDROID_HOME:$ANDROID_PLATFORM_TOOLS:$PATH"
  ```



Arribats a aquest punt ja podem crear la nostra primera APP de prova:

```
phonegap create app_prova
```

o bé

```
phonegap create app_prova com.exeple.app_prova aplicació
```

Aquesta comanda ens crea una carpeta `app_prova` on tenim un arbre d'arxius exemple a partir del qual començar a treballar. **A partir d'aquest moment totes les ordres del terminal de comendes s'han d'executar desde el directori principal del projecte que s'acaba de generar.**

A continuació podem entrar dins la carpeta i compilar mitjançant Phonegap:

```
phonegap build android
phonegap build ios
```

i a continuació executar l'aplicació sobre un dispositiu `-d` o bé un emulador `-e`:

```
phonegap install android
phonegap install ios
```

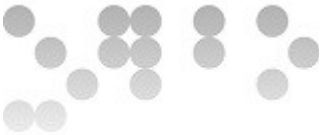
O bé fer ambdues passes de cop:

```
phonegap run -e android
phonegap run -e ios
```

És important insistir que cadascuna de les plataformes tindrà aspectes de configuració concrets de la plataforma que caldrà modificar.

Podem consultar:

http://docs.phonegap.com/en/edge/guide_platforms_index.md.html#Platform%20Guides



4.2.2 Workflows

Tal i com es comenta anteriorment Phonegap permet a partir de un mateix arbre de directoris generar el codi font per a diverses plataformes.

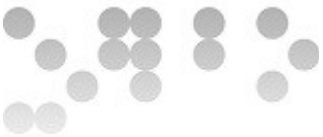
Per fer-ho es poden seguir dos camins o *workflows* a l'hora de desenvolupar. Tant l'un com l'altre són útils per tal d'arribar al mateix punt però és important saber quins avantatges i inconvenients ens ofereix cadascun per tal de saber quin és el més adequat a l'hora de treballar.

- **Cross-platform workflow:** Seguirem aquest fluxe de treball si volem desenvolupar per diverses plataformes alhora. Aquest fluxe esta centrat en la utilitat de Phonegap desde la línia de comandes com a eina d'alt nivell. El que fa aquest fluxe és, a partir d'uns arxius comuns, genera uns subdirectoris concrets per a cada plataforma fent els canvis necessaris per cadascuna de les plataformes i generant els binaris.

  És a dir que es modifica el codi principal generat a partir de `phonegap create`, i posteriorment es genera un subdirectori dins la carpeta `platforms` que conté el arbre de directoris amb el codi específic de cada projecte a partir de la comanda `phonegap build`.

- **Platform-centered workflow:** Seguirem aquest fluxe si volem desenvolupar per a una plataforma en concret i volem poder modificar-la abaix nivell. Faríem servir aquest camí, per exemple, si volguéssim barrejar components nadius amb components basats en Phonegap. Aquest fluxe de treball es basa en un conjunt d'arxius de baix nivell adaptats a cada plataforma i una eina separada que ens permet aplicar *plugins*. Òbviament aquest sistema proporciona un accés més ampli a opcions de desenvolupament concretes de la plataforma per a la que desenvolupem i ens permet treballar de manera híbrida (plataforma nadiva+Phonegap).

  És a dir que una vegada generat el directori específic per a la plataforma concreta modifiquem el codi específic d'aquesta plataforma.



Acostuma a ser més fàcil programar a partir de **cross-platform workflow** per generar una app i posteriorment passar a **platform-centered workflow** per tal d'afinar més en cadascuna de les plataformes, ajustant el resultat final.

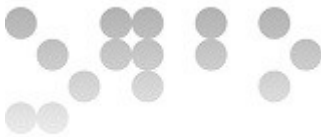
Aquest és un aspecte molt important i que s'ha de tenir molt en compte ja que una vegada passem de treballar de **cross-platform** a **platform-centered**, ja no podem tornar enrere perquè que en compilar, Phonegap generaría de nou l'arbre de directoris específic i ens matxacaria totes les modificacions fetes sobre la branca en qüestió.

4.2.3 Elecció de l'entorn

Com per a tots els llenguatges de programació, disposem d'eines integrades que ens poden facilitar força la feina, els anomenats Entorns Integrats de Desenvolupament o IDEs en el seu acrònim anglès. Entre els més populars trobem:

- Eclipse: Un dels entorns integrats per excel·lència. Esta llicenciat sota llicència de codi obert i per tant el podem descarregar lliurement sense cap cost. Cal afegir també que és tremendament versàtil i permet programar en multitud de llenguatges i integrar-hi multitud d'eines per adaptar-lo.
- ADT: Aquest entorn no és més que una versió d'Eclipse preparada per l'equip desenvolupador d'Àndroid amb totes les eines integrades i configurades per poder treballar: compilador, emulador, etc.
- Xcode: Aquest és l'entorn de programació d'Apple i seguint la política d'aquesta empresa només es pot executar en el seu SO propietari. Es gratuït per als usuaris d'Apple.
- Visual Studio: Aquest entorn està desenvolupat per Microsoft i com la majoria de software privatiu desenvolupat per aquesta empresa és de pagament. Esta en versions PC i Mac.

Tenint en compte l'objecte i el context en que es desenvolupa aquest projecte podria semblar que el més adient seria escollir algun dels entorns de codi lliure com l'ADT d'Android. I tot i que es comentaran els principals aspectes per fer-ho adequadament,



atès que Phonegap ens permet generar indistintament aplicacions per diverses plataformes en base a un mateix codi, sovint hem recorregut a Xcode i el seu emulador per desenvolupar, depurar i emular l'aplicació per motius d'eficiència, ja que la simulació i la depuració de codi en Android és brutalment lenta i ineficient en comparació amb aquest entorn. De retruc hem pogut apreciar quines diferències hi ha entre les dues plataformes i els avantatges i inconvenients de cadascuna.

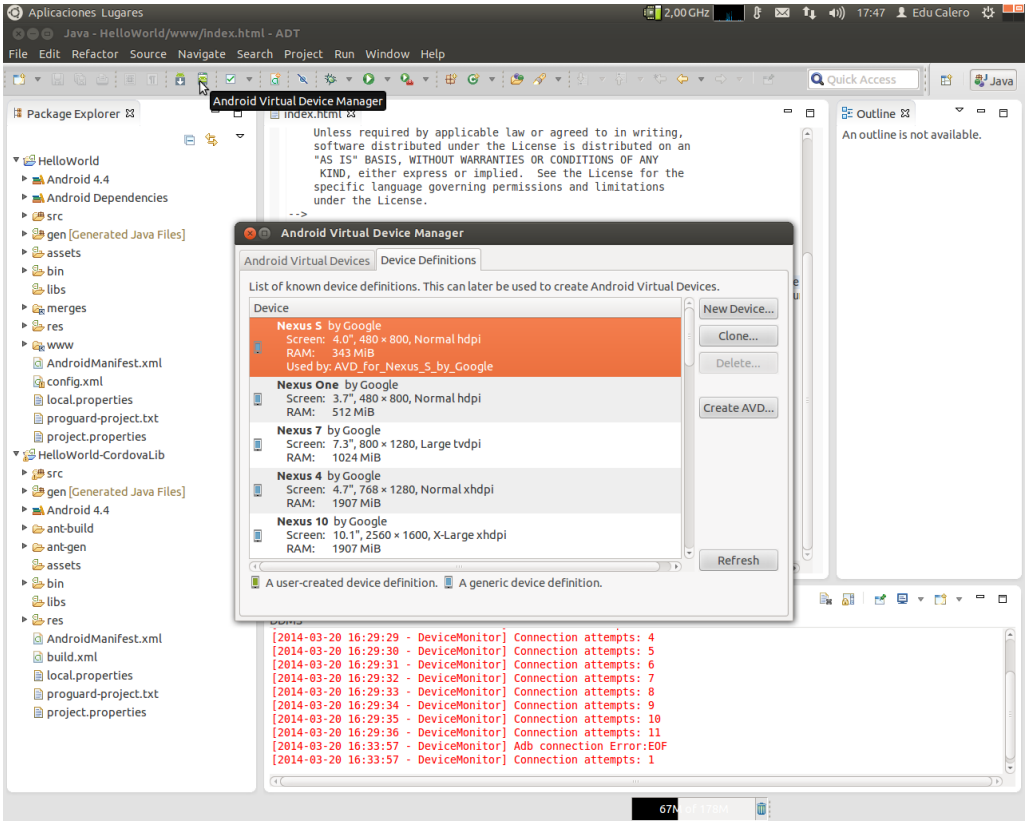
4.2.4 Preparació de l'AVD (Emulador)

Donem per suposat s'ha descarregat l'ADT a la web d'Android.

Cal a continuació instanciar un emulador sobre el qual es podrà executar i provar el codi compilat.

Gràcies a l'ADT, que integra l'emulació en Android per defecte, podrem instanciar un emulador perquè quan fem un `run` a phonegap desde el terminal de comandes, aquest sigui cridat i puguem veure el resultat. En cas contrari donarà error.

Per fer-ho obrirem ADT i clicarem sobre l'icona `AVD manager`, escollirem un dispositiu, com per exemple `Nexus S` i seguidament clickarem `Create AVD`, per crear una instància del dispositiu virtual sobre la qual podrem executar:



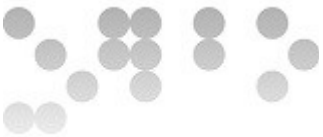
Un cop configurades les preferències ja tindrem preparat el nostre emulador.

4.2.5 Afegint *plugins*

Quan es compila un nou projecte, l'aplicació per defecte no permet gaires funcionalitats i perquè es pugui comunicar estretament i de forma nadiva amb els elements del dispositiu com ara la càmera o la memòria, caldrà afegir *plugins.* Aquests plugins permeten a partir d'una API controlar aquests elements del dispositiu de manera que són el pont de comunicació entre el codi Javascript i el dispositiu.

El repositori oficial de plugins de Phonegap es troba a:

https://build.phonegap.com/plugins.



Es poden afegir plugins de manera local o indicant un repositori:

```
$ phonegap local plugin add /path/to/plugin
$ phonegap plugin add https://github.com/apache/plugin.git
```

Per saber els plugins que tenim instal·lats:

```
$ phonegap plugin list
```

Per eliminar un plugin:

```
$ phonegap plugin remove org.apache.cordova.device
```

4.2.6 Config.xml

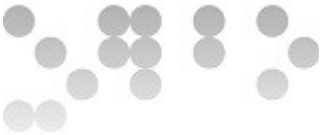
Molts dels aspectes de comportament de l'aplicació es controlen desde aquest arxiu de configuració global.

Quan generem directoris específics per a cada plataforma, aquest arxiu es copia en cadascun dels directoris, per tant, hi haurà paràmetres de configuració global que es faran servir per a *cross-platformworkflow* i d'altres específics de cada plataforma que es faran servir en *platform-centered workflow.*

Elements de configuració global

Es tracta d'element de configuració bàsics i metadades de l'aplicació com ara informació xml, el nom de l'aplicació, dades del desenvolupador, etc.

```
<widget xmlns     = "http://www.w3.org/ns/widgets"
        xmlns:gap = "http://phonegap.com/ns/1.0"
        id        = "com.phonegap.opendomo"
        version   = "1.0.0">

    <name>Opendomo_app</name>
```

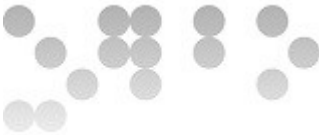


```
<description>
    Opendomo_app lets you control remotely your opendomo device.
</description>

<author email="educalerorovira@gmail.com">
    Eduard Calero Rovira
</author>
```

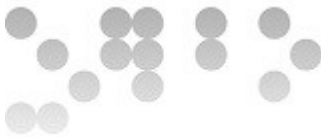
Preferències

Hi ha dos tipus de preferències: globals i multiplataforma. Les globals s'apliquen a totes les plataformes i les multiplataforma s'apliquen a algunes però no a totes.

```
<preference name="permissions"   value="none"/>
<!-- Customize your app and platform with the preference element. -->

<!-- <preference name="phonegap-version"      value="3.4.0" /> <!--
all: current version of PhoneGap -->

<preference name="orientation" value="default" />
<!-- all: default means both landscape and portrait are enabled -->
<preference name="target-device"              value="universal" />
<!-- all: possible values handset, tablet, or universal -->

<preference name="fullscreen"  value="true" />
<!-- all: hides the status bar at the top of the screen -->

<preference name="webviewbounce"  value="false" />
<!-- ios: control whether the screen 'bounces' when scrolled beyond
the top →

<preference name="prerendered-icon"   value="true" />
```



```
<!-- ios: if icon is prerendered, iOS will not apply it's gloss to the
app's icon on the user's home screen -->

<preference name="stay-in-webview"   value="false" />
<!-- ios: external links should open in the default browser, 'true'
would use the webview the app lives in →

<preference name="ios-statusbarstyle"   value="black-opaque" />
<!-- ios: black-translucent will appear black because the PhoneGap
webview doesn't go beneath the status bar →

<preference name="detect-data-types"   value="true" />
<!-- ios: controls whether data types (such as phone no. and dates)
are automatically turned into links by the system →

<preference name="exit-on-suspend"   value="false" />
<!-- ios: if set to true, app will terminate when home button is
pressed →

<preference name="show-splash-screen-spinner" value="true" />
<!-- ios: if set to false, the spinner won't appear on the splash
screen during app loading →

<preference name="auto-hide-splash-screen" value="true" />
<!-- ios: if set to false, the splash screen must be hidden using a
JavaScript API →

<preference name="disable-cursor"  value="false" />
<!-- blackberry: prevents a mouse-icon/cursor from being displayed on
the app →

<preference name="android-minSdkVersion"      value="7" />
<!-- android: MIN SDK version supported on the target device. MAX
version is blank by default. →
```



```
<preference name="android-installLocation"    value="auto" />
<!-- android: app install location. 'auto' will choose. 'internalOnly'
is device memory. 'preferExternal' is SDCard. -->
```

Elements de funcionalitat

Aquests elements s'apliquen a l'arxiu *config.xml* específic de cada plataforma fent servir el tag `<feature>`. Amb aquest tag s'habiliten les APIs a nivell de dispositiu (específiques del dispositiu) i els plugins externs:

```
<!-- Enable individual API permissions here. The "device" permission
is required for the 'deviceready' event. -->
<feature name="http://api.phonegap.com/1.0/device" />
```

4.3 Implementació

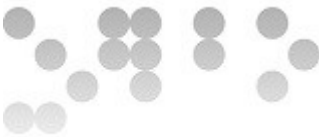
Es parteix del model i l'arbre de directoris inicialment generat per Phonegap.

La primera part a implementar és l'estructura del document en HTML al que posteriorment afegirem estil i dinamisme amb CSS i Javascript.

4.3.1 Estructura HTML

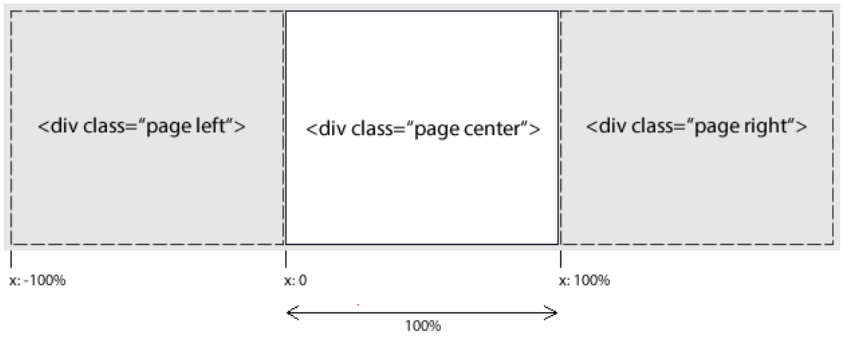
Per implementar aquesta primera part modificarem l'arxiu *index.html.*

Tal i com es pot apreciar més endavant al codi, aquestes pantalles que inicialment estimem estan representades pels contenidors*:* `menu_lateral`, `settings`, `menu_principal` *i* `cargando`*.* La pantalla `menu_principal` contindrà inicialment el *login* on es podrà introduir el nom d'usuari i el password i d'altres dades que ODControl ens requereix per tal de poder iniciar una sessió. Les pantalles `menu_lateral` *i* `settings` contindran menús i preferències respectivament. I `cargando` serà una pantalla amb un *gif* d'espera per les càrregues de dades.



Cadascun d'aquests contenidors representarà una pantalla, però com cadascuna d'elles tindrà una mida igual a la pantalla del dispositiu, només tindrem una visible i la resta quedaran una sobre l'altra o en posicions fora de la pantalla del nostre dispositiu, o el que és el mateix, invisibles. Serem nosaltres els qui mostrarem o no quan ens convingui en cada cas.

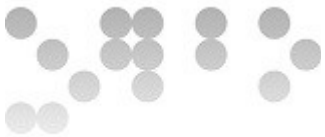
La idea és una cosa semblant a la següent:



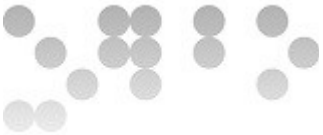
Cal tenir en compte que a l'hora d'implementar el codi, si no ho indiquem de manera explícita els *divs* es crearan de dalt abaix, el que vol dir que si es superposen, l'ultim quedarà al damunt de tot.

L'esquelet inicial serà doncs el següent:

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8" />
        <meta name="format-detection" content="telephone=no" />
        <!-- WARNING: for iOS 7, remove the width=device-width and
height=device-height attributes. See
https://issues.apache.org/jira/browse/CB-4323 →
```



```
        <meta name="viewport" content="user-scalable=no, initial-
scale=1, maximum-scale=1, minimum-scale=1, width=device-width,
height=device-height, target-densitydpi=device-dpi" />

<link rel="stylesheet" type="text/css" href="css/estilos.css" />
        <title> Opendomo </title>
    </head>

    <body>


<!-- Primera pantalla no visible dónde introduciremos el menú lateral
-->

       <div id="menu_lateral">

                <nav id="menu">

                    <a></a>
                    <a></a>
                    <a></a>

                </nav>

    </div>

<!-- Segunda pantalla no visible prevista para contener un panel de
control de configuraciones -->

    <div id="settings"> </div>

<!-- Tercera pantalla que contiene el panel de login -->
```

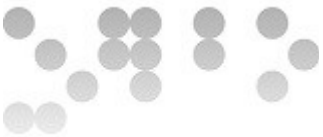


```
    <div id="menu_principal"> </div>

<!-- Pantalla que contiene un gif animado para mostrar en los momentos
de carga de datos -->

        </div> <div id="cargando">

            <div>
                <img src="img/ajax-loader2.gif" alt="Barra de carga"
height="60" width="60">
                    </div>

        </div>

<script type="text/javascript" src="phonegap.js"></script>
<script type="text/javascript" src="js/index.js"></script>

    </body>
</html>
```

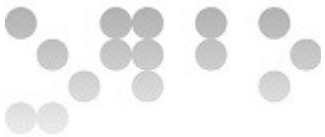
El contenidor corresponent a `menu_principal` contindrà inicialment dos botons representats per dues imatges a la part superior dreta i esquerra d'aquesta pantalla que ens permetran accedir als menús que apareixeran de manera lliscant dels laterals.

Per això dividirem aquesta pantalla en 3 parts: una capçalera superior que contingui els botons junt amb el logo d'Opendomo al centre, un cos que contindrà un contenidor amb el formulari d'accés i un peu que ens servirà per mostrar els retorns de l'aplicació:

```
<header id="cabecera">

<a href="javascript: menu('derecha');"><img class="btniz"
src="img/botonmenu.png"  alt="menubtn" height="30" width="30"> </a>
```



```
<img class="btncnt"src="img/logoopen.png" alt="Opendomo"  height="30"
width="150" >

<a href="javascript: menu('izquierda');"><img
class="btnder"src="img/settings.png"  alt="settingsbtn" height="30"
width="30"> </a>

          </header>

  <div id="cuerpo">
        <section id="loginbox">
            <form name="login_form


<p> User:
<input type="text" name="user" placeholder=" nombre de usuario "
required> </p> </BR>

<p> Password:
<input type="password" name="password" required> </p> <BR>

<input type="submit" value="Login" >

              </form>
         </section>

            <footer id="pie">

            </footer>
```



4.3.2 Primer format en CSS

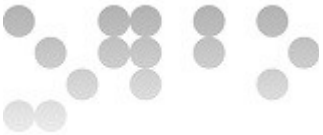
Un cop tenim el nostre esquelet inicial cal donar forma a aquest esquelet, i també implementar gràcies a CSS3 les classes que ens permetran implementar els movimients dels menús (*estilos.css*):

```
*{
     margin: 0px;
     padding: 0px;
     }

body {
     font-family:'HelveticaNeue-Light', 'HelveticaNeue', Helvetica,
Arial, sans-serif;
     height:100%;
   width:100%;
     position: fixed;
}
```

Primerament fem servir body com a contenidor general de la resta de contenidors i li donem una alçada i amplada igual a tota la pantalla del dispositiu. Això ens servirà de referència per la resta de contenidors que quedaran al seu interior.

Com es pot veure a continuació tots els contenidors fills directes, que són les 4 pantalles abans comentades, quedaran dins de body i cobriran tota la seva extensió.

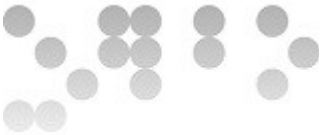
```
body > div {
     position: absolute;
     width: 100%;
     height: 100%;
     }

#menu_principal {
```



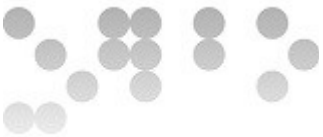
```
    background-color: #999999;
    }
```

En quest punt donarem format a alguns dels tres contenidors que posicionarem dins del contenidor *menu_prinicpal: cabecera, cuerpo, footer.*

```
.cabecera {
    position:absolute;
    top:0px; left:0;
    background-color:white;
   width:100%;
   height: 40px;
   border-bottom: 10px solid #043E95;
   text-align: center;
    }

#cuerpo{
    position: absolute;
    top: 50px;
    left: 0;
    width: 100%;
    height: 75%;
    text-align: center;
}

#loginbox{
    display: block;
    position: absolute;
    border: 2px solid #043E95;
    width: 40%;
    padding: 20px;
    margin: 20px 22.5%;
```



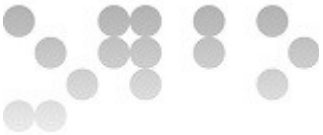
```
    background-color: white;

    -moz-border-radius: 5px;
     -webkit-border-radius: 5px;
     border-radius: 5px;
    }

#pie{
    display: block;
    position: absolute;
    z-index : 2;
    border-top:2px solid #043E95;
    width:100%;
    height: 100%;
    top: 85%;
    background-color: white;
    }

#cargando > div {
    z-index: 4;
    position: absolute;
    top: 40%;
    left: 40%;
    }

.btniz { float: left; display: none;  }
.btnder { float: right; display: none; }
.btncnt { padding-left: 10px; }
.btn{
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    border-radius: 5px;
}
```



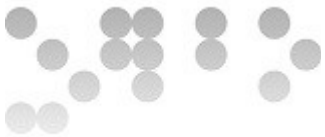
A partir d'aquí declararem unes classes que són les que farem servir per generar els moviments de les pantalles de manera lliscant (*sliding windows*).

Una classe principal *.page* que ens posicionarà les capes centrades en la pantalla del nostra dispositiu i la resta de classes a partir de la propietat *translate*, que en afegir-la mitjançant Javascript, ens mouran les pantalles a dreta i esquerra la proporció que desitjada.

```
/* Estilo de capa general */

.page {

position: absolute;
top: 0;
left: 0;
width: 100%;
height: 100%;

-webkit-transform: translate3d(0, 0, 0);
transform: translate3d(0, 0, 0);
}

/* Estilo de capa izquierda */

.page.left {
-webkit-transform: translate3d(-90%, 0, 0);
transform: translate3d(-90%, 0, 0);
}

/* Estilo de capa central */

.page.center {
-webkit-transform: translate3d(0, 0, 0);
transform: translate3d(0, 0, 0);
```



```
}

/* Estilo de capa derecha */

.page.right {
-webkit-transform: translate3d(90%, 0, 0);
transform: translate3d(90%, 0, 0);
}


/* Estilo de capa totalmente a la izquierda */

.page.totalleft {
-webkit-transform: translate3d(-100%, 0, 0);
transform: translate3d(-100%, 0, 0);
}

/* Estilo de capa totalmente a la derecha */

.page.totalright {
-webkit-transform: translate3d(100%, 0, 0);
transform: translate3d(100%, 0, 0);
}

/* Estilo de animación general */

.page.transition {
-webkit-transition-duration: .15s;
transition-duration: .15s;
}
```



4.3.3 Implementant moviments en Javascript

Gràcies a les classes que acabem d'implementar obtindrem un moviment en modificar dinàmicament la classe de cadascun dels contenidors. Per fer-ho caldrà que els botons cridin una funció Javascript que serà l'encarregada de fer aquesta modificació.

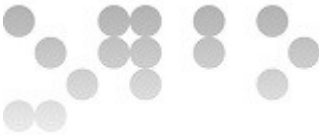
Seguint l'arbre de directoris que ens ha generat Phonegap implementarem de nou el fitxer `index.js` com a fitxer principal Javascript, és a dir, que tenint en compte l'ordre seqüencial de lectura del codi a l'hora d'ésser interpretat, aquest serà el primer fitxer llegit de tota l'aplicació en aquest llenguatge. Aquest és un aspecte que caldrà tenir molt present, ja que abans de treballar amb una variable o cridar una funció, cal que estigui carregada en memòria.

És per això que el primer que farem es declarar cadascun dels objectes amb els que treballarem referenciats per la seva `id`.

```
// Declaraciónn de variables globales
var estado, menulateral, menuprincipal, settings, cargando;

// Guardamos en variables elementos para poder rescatarlos después sin
tener que volver a buscarlos
 menuprincipal = document.getElementById("menu_principal"),
 menulateral = document.getElementById("menu_lateral"),
 settings = document.getElementById("settings"),
 cargando = document.getElementById("cargando");
```

Quan treballem amb Javascript el software que interpreta el codi genera una sèrie d'events per comunicar-nos estats. Per exemple, quan es carreguen certs elements del DOM (*Document Object Model*), com ara *window* que és l'objecte pare de tota la resta, ho comunica amb un event `load`.

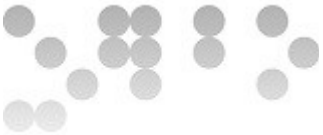
Això vol dir que a través d'aquests events es poden cridar pedaços de codi en moments concrets sense que siguin executats en el moment de càrrega a la memòria



o el que es el mateix, son funcions disparades en el moment que succeeix l'event.

Tenint en compte això, el que expressa el següent codi és que fins que no tinguem carregada la finestra, no començarem a modificar les classes dels elements per posicionar-los a la pantalla (o fora).

```
window.onload = function()
{
     estado = "menuprincipal";

     // Añadimos las clases necesarias
     menuprincipal.className = 'page center';
     menulateral.className = 'page center';
     settings.className = 'page center';
     cargando.className = 'page totalleft';

}


     function menu(opcion)
     {

     // Si pulsamos en el botón de "menu" entramos en el if
     if(opcion=="derecha")
     {

          if(estado=="menuprincipal")
          {
               menuprincipal.className = 'page transition right';
               settings.className = 'page transition right';
               estado="menulateral";

          } else if(estado=="menulateral")
          {
```



```
            menuprincipal.className = 'page transition center';
            settings.className = 'page transition center';
            estado="menuprincipal";
        }

    // Si pulsamos el botón settings entramos en el elseif

    } else if(opcion=="izquierda")

    {
            if(estado=="menuprincipal")
            {
            menuprincipal.className = 'page transition left';
            menulateral.className = 'page transition left';
            estado="settings"

            } else if(estado=="settings")
            {
            menuprincipal.className = 'page transition center'
            menulateral.className = 'page transition center';
            estado="menuprincipal";
            }
    }
}
```

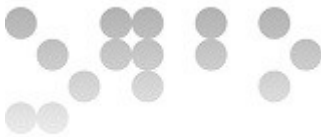
Finalment ja podem implementar la funció que generarà el moviment. El que obtindrem és una cosa semblant a la següent:

4.4 XMLHttpRequest

XMLXttpRequest és l'eina que ens permet implementar crides HTTP i HTTPS a servidors web.

És doncs aquesta interfície a través de Javascript la principal eina de la que disposarem per tal de poder-nos comunicar amb el dispositiu ODControl a través del port corresponent i de consultar altres arxius dins el propi arbre de directoris per tal de refrescar informació del contingut de la nostra aplicació sense haver de recarregar de nou tot el contingut de la pàgina (AJAX).

En Javascript aquesta API està implementada a través d'un objecte al que cal instanciar i configurar per tal de fer cadascuna de les crides que siguin necessàries. Per cada petició caldrà instanciar un objecte XMLHttpRequest.



4.4.1 Cross Origin Resource Sharing (CORS)

CORS és una tècnica utilitzada per permetre l'accés a través de javascript amb l'objecte XMLHttpRequest, a recursos remots d'altres dominis de manera segura.

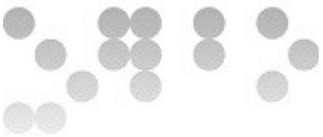
El concepte de *política del mateix orígen* (en *anglès same-origin policy)* és un concepte que es deriva del *model de seguretat d'aplicacions web* (en anglès web application security model) que només permet accedir al *DOM a scripts* del mateix domini, per que aquest sigui modificat.

La idea bàsica de CORS és fer servir els encapçalaments de les peticions tant de servidor com del client (normalment un navegador) per valorar si es pot o no accedir als recursos.

Per indicar quins dominis poden accedir als nostres recursos caldrà modificar primerament el següent paràmetre a *config.xml:*

```
<!--
Define access to external domains.

<access /> - a blank access tag denies access to all external
resources.

<access origin="*" /> - a wildcard access tag allows access to all
external resource.

Otherwise, you can specify specific domains:
    -->
<access origin="*" />

<!--<access origin="http://127.0.0.1*"/> allow local pages →
```

Inicialment nosaltres permetrem accés a qualsevol recurs desde l'exterior, tot i que, tal i com es pot llegir a les línies comentades podríem filar molt més prim i cenyir-nos, per exemple, a l'adreça o domini del nostre dispositiu per evitar ingerències.



Pel que fa a la part activa del nostre codi, caldrà afegir una sèrie de línies a les capçaleres de les nostres peticions per tal d'indicar que nosaltres només rebrem recursos del domini del nostre dispositiu, i de retruc aprofitarem també per indicar que per accedir als recursos caldrà introduir credencials (i també afegirem quines són).

Les línies de capçalera que caldrà afegir a les nostres peticions són:

```
Access-Control-Allow-Origin: url
Authorization: Basic  user:password
```

On

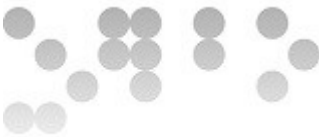
- `url` és el domini del dispositiu.
- `user` és el nom d'usuari.
- `password` és el password.

4.4.2 Funció Request: *REQ(cmd)*

Ara que ja sabem com fer servir l'XMLHttpRequest, caldrà implentar una funció que serà a través de la qual farem les nostres peticions al dispositiu.

La funció rebrà un argument comanda (*cmd*) i retornarà un objecte resposta a partir del qual podrem extreure les dades que ens contesti la nostra petició.

```
//Función request
function REQ(cmd){
var req = new XMLHttpRequest();

//true > asíncrono (continúa); false > síncrono (espera la respuesta);
req.open("GET", url+cmd, true);

//Se añaden las siguientes cabeceras que permiten al navegador no ser
afectado por la política del mismo orígen y enviar las credenciales.
```



```
req.setRequestHeader("Access-Control-Allow-Origin", url);
req.setRequestHeader("Authorization", "Basic " + btoa(user + ":" +
password));


//Se manda la petición del recurso a través del objeto req.
req.send(null);
return req;
}
```

Per provar que la funció té un funcionament correcte, la cridem a través del botó *submit* de la pantalla de *login* enviant una comanda `ver+`.

4.5 Estructuració del codi

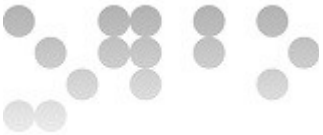
Fins al moment, hem vist com hem implementat l'estructura inicial i bàsica a través de la qual podem introduir les credencials i fer peticions al dispositiu.

Ara que el software ja es pot comunicar amb el dispositiu caldrà implementar el gruix de l'aplicació, però per fer-ho caldrà pensar com haurem d'estructurar a grans trets el codi javascript.

Tal i com s'ha fet desde el principi i seguint l'arbre de directoris entregat per Phonegap, el primer fitxer Javascript que serà cridat serà `index.js`.

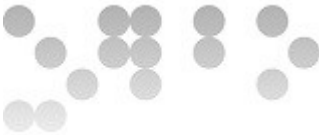
Dins d'aquest fitxer i seguint el que s'ha descrit fins al moment hi haurà una primera secció de declaració de variables globals i la seva referenciació. Seguidament en aquest fitxer declararem les funcions relacionades amb events que s'encarregaran d'inicialitzar l'aplicació i finalment declararem funcions troncals de l'aplicació que seràn cridades un cop ja tot estigui carregat.

D'altra banda i en segon terme es declararà un segon fitxer amb funcions secundàries anomenat `functions.js`. A aquest fitxer declararem en el seu inici una sèrie de funcions bàsiques que degut al seu volum d'ús durant la resta del codi, quedaran abreviades per tal d'accedir de manera més ràpida i còmode i de retruc, fer el codi molt més entenedor.



Són les següents:

```
//function getElementbyID
function ID(id){
    return document.getElementById(id); }

//Funcion GetelementbyTagname retrona un array de objetos para un Tag
dado
function GC(cls){
    return document.getElementsByClassName(cls); }

//Muestra texto por la consola inferior (pie) que se usa como log.
function log(t){pie.innerHTML = t;}

// Función SetAtribute. A un objeto n le añade un atributo a con un
valor v
function SA(n,a,v){ n.setAttribute(a,v);return n;}

//Función AppendChild: al objeto p le crea un hijo c
function AC(p,c){p.appendChild(c);}

//Crea un texto "t" que luego meterá mediante appenchild dentro del
elemento.
function CT(t){return D.createTextNode(t);}

//Funcion CreateElement crea un elemnto "t" con clase "c" e id "i"
function CE(t,c,i){
    var e=document.createElement(t);
    if(c&&c!='')SA(e,'class',c);
    if(i&&i!='')SA(e,'id',i);
    return e;
}
```



En base a aquestes abreviacions queden modificades totes les funcions que en consten en aquest llistat i s'havien fet servir fins al moment, per la seva versió abreviada.

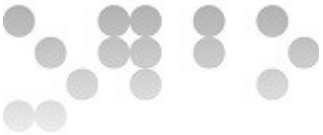
4.6 Accés a memòria del dispositiu

Per tal de carregar sobre l'aplicació el mapa sobre el qual dibuixarem els diferents interruptors, implementarem un botó a través del qual accedirem al carret del dispositiu i seleccionarem la imatge que volem sigui carregada.

```
<p> Image:

<input type="button" name="planos"
value="download"onclick="carrete()"> </p> <BR>

<p><input type="button" name="botonlogin"
   value="Login" onclick="logar()"></p>
```

En prémer el botó es dispararà la funció `carrete()`. Aquesta funció a partir del corresponent plugin de phonegap ens permetrà fer servir l'objecte `navigator.camera` que degurament configurat, accedirà a la memòria i ens permetrà seleccionar imatges emmagatzemades prèviament al dispositiu i ens retornarà la seva *url*, que posteriorment, encastarem a la pantalla:

```
//Cargar imágenes
function carrete()
{

 navigator.camera.getPicture(onSuccess, onFail, { quality:
90,destinationType: destinationType.FILE_URI, sourceType:
Camera.PictureSourceType.PHOTOLIBRARY });
```



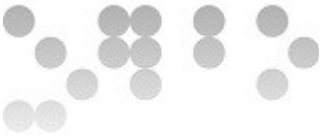
```
}


function onSuccess(imageURI) {

   cuerpo.innerHTML='<img id=miimagen src="'+imageURI+'">';

    }

    function onFail(message) {
        log('Failed because: ' + message);
    }
```

Tal i com es dedueix del codi, un cop finalitzada l'execució de la funció, es dispararà la funció *onSuccess* o *onFail*, si l'execució del mètode `getPictuture` ha tingut èxit o no.

La funció *onFail*, el que es fa és mostrar el corresponent missatge a través de la pantalla de log de la part inferior. Mentre que *onSuccess* implementa codi html que substitueix el contingut del contenidor `cuerpo` dins de la pantalla `menu_principal`, per la imatge que tinguem seleccionada. Això substituiria el contingut que fins al moment és el formulari de login.

Cal comentar també que per fer servir l'objecte `navigator.camera` cal inicialitzar unes variables. El punt adequat, és en el moment de inicialització del dispositiu. Per fer-ho cal modificar la funció que crida a l'event `deviceready` declarada a la segona secció del fitxer, just després de declarar i definir les variables globals `index.js`.

```
//Inicializamos la app
var app = {
initialize: function() {
    this.bindEvents();
},
```



```
bindEvents: function() {
    document.addEventListener('deviceready', this.onDeviceReady,
false);
},
onDeviceReady: function() {
    pictureSource=navigator.camera.PictureSourceType;
    destinationType=navigator.camera.DestinationType;
}
};
```

D'aquesta manera les variables `PictureSourceType i DestinationType` queden definides i llestes per ser utilitzades en el mètode `getPicture`.

4.7 Llibrería iScroll

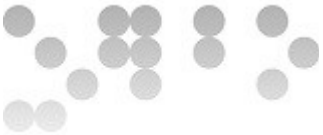
Tal i com esta definida la funció anterior, la imatge seleccionada es mostra de manera estàtica i parcial dins el contenidor degut a la seva resolució.

Un dels aspectes que ens trobem a l'hora de representar la imatge sobre la pantalla del dispositiu, és que sovint, la resolució del mapa és molt major que la del display del nostre dispositiu i, per tant, cal poder escalar-la d'alguna manera per tal de poder navegar sobre ella i interactuar-hi.

Per tal de donar resposta a aquest problema, ens servirem de la llibreria iScroll. Aquesta llibreria Javascript en les seves diverses versions, ens permet interactuar amb el dispositiu i a través dels moviments dels dits sobre la pantalla i fer un *zoom in, zoom out* o desplaçar-nos entre d'altres.

Per fer-la servir, només cal linkar-la al nostre codi principal (`index.html`) tenint en compte que en el moment del seu ús haurà d'estar carregada en memòria:

```
<script type="text/javascript" src="js/iscroll-zoom.js"></script>
```



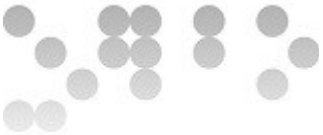
Un cop fet això caldrà declarar un contenidor amb *id=wrapper* i un altre amb *id=scroller.* El `scroller` simularà la finestra a través de la qual veurem la part de la imatge continguda al `wrapper`. És a dir, la idea serà com si agaféssim dos fulls, un sobre l'altre, i al de la part superior li retalléssim un quadre per veure el de sota amb la diferència que el full de la part inferior el podem escalar.

```
cuerpo.innerHTML='<div id="wrapper"> <div id="scroller"><img
id=miimagen src="'+imageURI+'"> </div></div>';

//Aplicamos la librería iscroll-zoom sobre el código HTML creado
        myScroll = new IScroll('#wrapper', {
                                zoom: true,
                                scrollX: true,
                                scrollY: true,
                                mouseWheel: true,
                                wheelAction: 'zoom'
                                });
        document.addEventListener('touchmove', function (e)
{ e.preventDefault(); }, false);}
```

Finalment tal i com es pot veure al codi, caldrà instanciar un objecte *iScroll* i configurar una sèrie de paràmetres per tal d'indicar el comportament de la finestra. En la nostra configuració permetrem el zoom, el scroll en ambdós eixos i el zoom.

4.8 Funció logar

L'aplicació ja és capaç d'enviar credencials i comunicar-se amb el dispositiu, carregar una imatge i poder navegar sobre ella. A continuació implementarem una millora en l'accés al mapa que donarà un millor aspecte al conjunt.



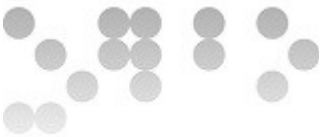
Donat que es pot donar el cas que la connexió de vegades no sigui tot el ràpid que voldríem, de vegades la comunicació entre l'aplicació i el dispositiu remot pot trigar alguns segons. Això pot donar sensació de que l'aplicació s'ha quedat penjada i no funciona adequadament. El que farem a continuació és tenint en compte certs events generats per l'objecte XMLHttpRequest, mostrar la pantalla de càrrega implementada anteriorment i que per defecte queda fora del camp visible de l'aplicació.

L'algoritme de la funció el que farà és mostrar la pantalla de càrrega en primera instancia i actualitzar la *consola de login* a la part inferior de la pantalla de l'aplicació mostrant missatges de retorn amb la resposta a les peticions. Posteriorment mostrarà els botons que ens permetran accedir a les pantalles de menú i configuració ubicats a la part superior dreta i esquerra de la pantalla de manera que en la pantalla inicial no podrem fer res excepte introduir les credencials i l'aplicació no quedarà operativa fins que s'hagi verificat que les dades són correctes.

```
function logar()
{
// Se muestra el gif de carga y en mensaje en la consola inferior
cargando.className = 'page center';
log("sending credentials... "+" user: "+user+" password: "+password+"
URL: "+url+" \n");

//Se muestran los botones
    botonder.style.display = "initial";
    botoniz.style.display = "initial";
```

Seguidament enviarà una ordre `lsc` al dispositiu mitjançant la funció request (REQ), per tal de comprovar que la connexió es correcta i de retruc mostrar la versió del software de que disposa el dispositiu al que enviarem les ordres. Ho mostrarà també a través de la finestra de log a la part inferior.

En cas de que la resposta sigui bona, és a dir, trobi correctament el recurs, retirarà la

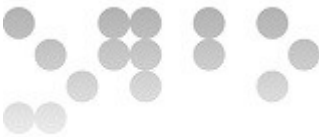


pantalla de càrrega i cridarà la funció *createMenu()* que com veurem més endavant s'encarregarà de generar dinàmicament el menú a partir de la resposta obtinguda a partir del qual podrem seleccionar els interruptors i sensors que després ubicarem sobre el mapa.

```
//Hace una consulta lsc y a partir de la respuesta crea el menu de
puertos.

    response = REQ("lsc+");

    response.onreadystatechange = function (){

    //Se muestra el resultado en la parte inferior de la pantalla
    log(response.readyState+" "+response.statusText+" \n");

if (response.readyState == 4){

    //Se retira el gif de carga
    cargando.className= 'page totalleft';


    if (response.status == 200)  {

    createMenu(response.responseText);
    response = REQ("ver+");

    log(response.responseText+" \n");
```

4.9 Funció createMenú

Aquesta és una funció força important dins del codi de l'aplicació. S'encarregarà de generar el menú de configuració a partir del qual seleccionarem elements que

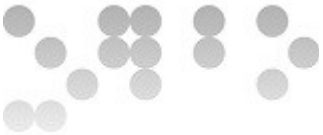


posteriorment ubicarem sobre el mapa, mantenint en tot moment actualitzat el menú d'acord amb les dades del dispositiu que alhora es comunica amb els sensors i interruptors de l'espai que controlarà.

Dins de la pantalla de configuració `menu_lateral`, que inicialment quedarà fora del display del dispositiu (a la part esquerra), es disposaran dos menús desplegables de tipus *`select`*. El primer, per tal d'inserir objectes al mapa i el segon per tal d'eliminar-ne. Inicialment les opcions estaran buides, però a partir de la resposta a la primera comanda *`lsc`* al dispositiu remot, aquest ens retornarà una relació en format text de tots els ports i el seu estat.

És en aquest punt on començarà a fer la seva tasca, escombrant el text retornat pel dispositiu i identificant el tipus de port i el seu estat i posteriorment generant dinàmicament una opció dins l'etiqueta *`select`* al menú de l'aplicació, perquè cadascun dels sensors o interruptors sigui seleccionable.

```
<div id="menu_lateral">
      <header class="cabecera">MENU</header>
           <nav id="menu">

            <table>

  <tr>
      <th colspan="3">Manage ports</th>
  </tr>

  <tr>
     <td><select name="port" id="opciones"></select> </td>
     <td colspan="2">
         <input type="button" value="Add button" onclick=
"addbutton()"></input>
     </td>
```



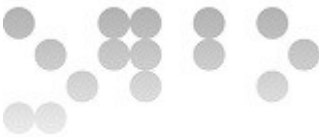
```
  </tr>

     <tr>
    <td><select name="port" id="removes"></select> </td>
    <td colspan="2"><input type="button" value="Remove button"
onclick="removebutton()"></input>
     </td>
   </tr>
```

Això en *Javascript* s'implementarà: primerament identificant els objectes `select` als quals afegirem les opcions, i posteriorment guardant en una variable la resposta del dispositiu que haurem de llegir per obtenir les dades que volem:

```
function createMenu(tx){

    d = ID("opciones");
    r = ID ("removes");


    var a = tx.split("\n");
```

Per entendre com llegirem les dades és important saber en quin format les rebrem. Per refrescar una mica la memòria observem la resposta a una comanda `lsc`:

```
DO000:DOM_:ON
do001:DOMl:OFF
do002:DOM_:ON
do003:DOMc:OFF
do004:DOM_:OFF
do005:DOM_:OFF
do006:DOM_:ON
do007:DOM_:OFF
di000:DIM_:OFF
di001:DIM_:OFF
```



```
di002:DIM_:OFF
di003:DIM_:OFF
di004:DIM_:OFF
di005:DIM_:OFF
di006:DIM_:OFF
di007:DIM_:OFF
ao000:AOM_:+0000.0040:+00000|00020:a5:00255
ao001:AOM_:+0000.0000:+00000|00020:a5:00255
ai000:AIM_:+0000.0360:+00000|00020:a5:00255
ai001:AIM_:+0000.0240:+00000|00020:a5:00255
vt000:AVM_:+0000.0040:PER:+0000.0040:00010
vt001:DVC_:OFF:PER:OFF
vt002:DVC_:OFF:VAR:OFF
AND03:DvMs:OFF:GRP:di000 di001 and
vt004:DVC_:OFF:VAR:OFF
vt005:DVC_:OFF:VAR:OFF
vt006:DVC_:OFF:VAR:OFF
vt007:DVC_:OFF:VAR:OFF
vt008:AVC_:+0010.0000:PER:+0010.0000:00010
vt009:DvC_:ON:GRP:DO000 DO000 and
vt010:XVC_:
... (ports virtuals)
vt061:XVC_:
$ON__:DVH_:ON:PER:ON
$OFF_:DVH_:OFF:PER:OFF
DONE
```

Tal i com queda més extensament explicat al capítol 4 (apartat 1.3), trobem cadascun dels ports, encapçalats pels 8 ports digitals de sortida(DO), 8 ports digitals d'entrada(DI), 2 ports analògics de sortida(AO), 2 ports analògics d'entrada(AI) i finalment, tota la resta fins arribar al `DONE`, que indica el final de les dades.

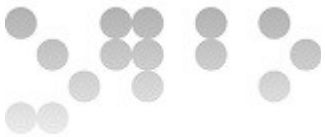
Els ports rellevants per a la nostra aplicació seran els 8DO, 8DI, 2AO i 2AI.

En el cas dels ports digitals el format és el següent:

`nom:DXM_:valor`

On:

- `X` serà `O` en els de sortida (interruptors) o `I` els d'entrada (sensors).



- Els `valor` possibles són ON i OFF.

En el cas dels ports analògics el format és el següent:

`nom:AXM_:valor:max|min:tol`

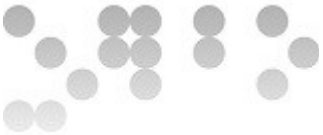
On:

- `X` serà `O` en els de sortida o `I` els d'entrada (sensors).
- `Valor` serà un valor entre `-9999,9999 i 9999,9999`.
- `max | min` és el rang de valors màxim i mínim que ens permetrà fer la conversió a unitats, per exemple graus centígrads (ºC)
- `tol` és el valor de tolerancia, és a dir per a canvis menors al valor no es notificarà canvi d'estat.

En resum, el que volem extraure de les dades que ens retornarà el dispositiu, seran els 3 primer camps: nom, tipus i valor.

Per tal de fer-ho, un cop tenim la resposta escombrem el resultat element a element amb un bucle *for* de principi a final. Donat que els camps es divideixen en 3 parts, pel signe *':' (dos punts)* cal anar *trencant* cadascuna de les línies a través del mètode `split`, que alhora processarem per tal d'obtenir només els ports que volem (DO, DI, AO i AI):

```
    for (var i=0; i<a.length; i++) {
        b=a[i].split(":");
        if (a[i] != "DONE")
            //OP tiene que devolver elementos option que va añadiendo
            try {AC(d,OP(b[0],b[1].substring(0,2),b[2]));} catch(e){}
    }
}
```

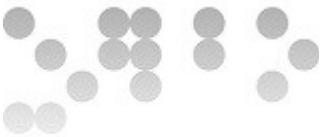


```
//Esta función crea una opción
    function OP (nom, tipo, valor){

        if (valor){
            if (tipo == ("DI")|| tipo == ("DO")||tipo == ("AI") ||
tipo == ("AO") )
            {

            var no = CE("option");
            no.text = nom;
            no.tipo = tipo;
            no.value = valor;
            no.disabled = false;
            return no;
            }
        }
    }
```

A mida que anem obtenint cadascuna de les línies de resposta i que sabem que no és la darrera (`DONE`), passem a la funció opció (`OP`) 3 atributs: el primer camp (`nom`), les dues primeres lletres del segon camp (`tipo`) i el tercer camp (`valor`). Aquesta funció **SI** el camp valor no està buït i **SI** el tipus és `DI`, `DO`, `AI` o `AO`, crea un element de tipus `option` amb una sèrie d'atributs que és retornat i afegit com a element fill de l'objecte select `opciones` (`d`), creant un llistat desplegable amb tots els ports analògics i digitals d'entrada i sortida.

4.10 Funció addbutton

Un cop generat el llistat caldrà implementar un sistema que ens permeti carregar sobre el mapa els diferents ports. És en aquest punt on entrarà en joc la funció que comentarem a continuació que serà cridada desde el botó que tindrem just al costat

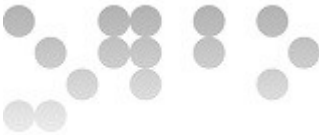


del llistat amb els ports.

El que fa aquesta funció és extreure les dades dels atributs de l'opció que tinguem seleccionada en el moment de prémer el boto '*afegir interruptor*' al mapa. A partir d'aquestes dades tractarà de buscar si hi ha un objecte amb una ID que es correspongui amb el nom de l'interruptor que volem afegir, ja que això voldria dir que prèviament ja s'havia afegit aquest interruptor i, per tant, no cal crear-ne un de nou sino re-ubicar el que ja existeix.

```
function addbutton(){
      imagen = ID("scroller");

//d.selectedIndex hace referencia al elemento selecionado en el
"select".

//Busca que no exista un elemento con ID igual al campo "text" del
elemento selecionado.
       if (!ID(d.options[d.selectedIndex].text)){
    ...

    ...
        }
        else {
            object = ID(d.options[d.selectedIndex].text);
            object.style.display = 'block';

        }
```

En cas de tractar-se d'un element nou, a partir del camp `tipo` podrem saber de quin tipus es tracta i a partir d'aquí tractar-lo en funció de si és `DO`, `DI`, `AO` o `AI`, fent servir un `case`.



```
if (!ID(d.options[d.selectedIndex].text)){
        switch (d.options[d.selectedIndex].tipo){
```

```
case "DO":
```

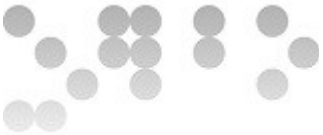
En aquest cas només cal crear un objecte `div` amb una clase `onswich_o` o `offswich_o` (que definirà la imatge que representarà l'interruptor com un interruptor obert o tancat) i ID igual al camp `text (nom)`.

Aquest objecte es crearà però no es mostrarà (`display=none`) sino que el mostrarem posteriorment quan l'usuari indiqui sobre el mapa mantenint apretat el dit, on el vol ubicar. Per tant caldrà afegir l'element creat com a node fill del contenidor de la imatge amb el mapa.

```
if (d.options[d.selectedIndex].value == "ON") {
                    object =
CE('div','onswitch_o',d.options[d.selectedIndex].text);
                    object.style.display = 'none';
                    AC(imagen,object);
                 }
                if (d.options[d.selectedIndex].value == "OFF"){
                    object =
CE('div','offswitch_o',d.options[d.selectedIndex].text);
                    object.style.display = 'none';
                    AC(imagen,object);
                }
```

En aquest cas concret DO, com el que estem creant són interruptors, caldrà també implementar dos disparadors d'events que quedaran explicats més endavant, a partir dels quals podrem encendre o apagar els interruptors.

```
   object.addEventListener('touchstart', tapandhold, false);
                object.addEventListener('touchend', function(){ hold =
```



```
false; clearTimeout(cuenta); }, false);
```

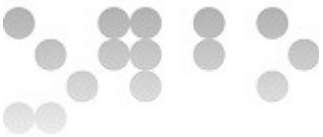
```
case "DI":
```

Aquest cas és exactament igual al cas DO però encara més senzill, ja que al tractar-se de sensors, no caldrà crear cap disparador sino que seran les dades del dispositiu recollides dels sensors les que ens donaran ordre de modificar el seu estat a partir de lectures que farem cada cert temps mostrant unes imatges amb bombetes enceses o apagades.

```
if (d.options[d.selectedIndex].value == "ON") {
                    object =
CE('div','onswitch_i',d.options[d.selectedIndex].text);
                    object.style.display = 'none';
                    AC(imagen,object);
                }
                 if (d.options[d.selectedIndex].value == "OFF"){
                    object =
CE('div','offswitch_i',d.options[d.selectedIndex].text);
                    object.style.display = 'none';
                    AC(imagen,object);
                }break;
```

```
case "AO":
```

En aquest cas es generaran objectes amb una classe que definirà un input al que poder donar un valor d'entrada. Es declara un disparador que crida a una funció que envia les dades al dispositiu cada vegada que es modifica el seu valor.



```
object = CE('input','analogswitch_o',d.options[d.selectedIndex].text);
                object.style.display = 'none';
                object.addEventListener('change', refresh, false);
                AC(imagen,object);
                object.value = d.value;
                break;
```

```
case "AI":
```

En aquest cas es generaran enlloc de bombetes o interruptors, objectes amb una classe que definirà un contenidor a través del qual es mostrarà un valor retornat pel sensor.

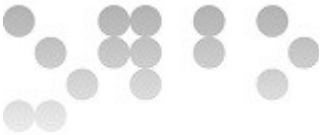
```
object = CE('div','analogswitch_i',d.options[d.selectedIndex].text);
                object.style.display = 'none';
                AC(imagen,object);
                object.innerHTML = d.value;
                break;
```

Finalment una vegada creat l'objecte i situat com a node fill del contenidor del mapa caldrà convidar a l'usuari a indicar-nos on cal situar-lo. El que farem per que sigui intuïtiu és mostrar automàticament el mapa i mostrar un missatge:

```
menu('derecha');
log("Hold your finger 2sec to place the button");
```

Aleshores quan l'usuari mantingui el dit més d'un temps X (que podrem modificar) sobre el mapa, quedarà situada la representació del port.

Per aconseguir-ho ens servirem de tres disparadors sobre el contenidor del mapa, dels



quals veurem el seu funcionament més endavant.

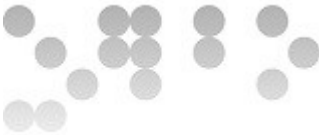
```
imagen.addEventListener('touchstart', tapandhold, false);
imagen.addEventListener('touchstart', getMousePosition, false);
imagen.addEventListener('touchend', holding = function(){ hold = false;
clearTimeout(cuenta); }, false);
```

4.11 Funció tapandhold(e)

Si observem l'apartat anterior veurem que hi ha uns disparadors, tant en els elements `DO` com al contenidor del mapa (`scroller`) que criden aquesta funció quan detecten que algú a començat a prémer sobre ells (`ontouchstart`).

El que fa la funció no és més que comprovar que, efectivament, algú esta polsant sobre ells amb un sol dit i durant més de X segons. En cas de que així sigui, la funció comprova sobre quin element s'esta polsant (`DO` o `imatge`) i determina quina acció s'ha de fer.

```
function tapandhold(e){

if (e.touches.length == 1){hold = true;} else {hold = false;}

cuenta = setTimeout (function(){

        if (hold == true){

        if (e.target.className == "onswitch_o" || e.target.className ==
"offswitch_o"){changeState(e.target); refresh();}

         else {placeButton();}

                  }},2000);
```



```
}
```

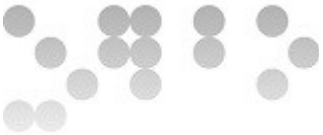
Si analitzem detingudament el codi veurem que hi ha una variable anomenada `hold` que és amb la que treballarem per saber si es manté o no polsat durant un cert temps l'element en qüestió.

De fet, el que fa el primer `if`, és comprovar si l'element que ha disparat la funció ha estat polsat amb un dit o més a través del mètode `touches.lenght`. Això es fa perquè es pot donar que intentem fer un zoom i es confongui aquesta acció amb la de ubicar un element sobre el mapa. En cas de ser així considerem que no s'esta mantenint el dit sino que l'acció és una altra i avaluem la variable `hold` a `false`. Altrament la variable s'avalua a `true` i l'execució continua.

Seguidament s'executa la funció `setTimeout` que s'emmagatzema dins la variable global `cuenta`. Aquesta funció ens permet definir codi que serà executat després de comptar fins a *X* milisegons (en l'exemple son 2000ms = 2s). És a dir, que si posteriorment a comptar 2s (en el cas de l'exemple), encara la variable `hold` esta avaluada a `true`, es seguiran una sèrie d'accions. En el cas de que la funció sigui disparada per un DO seran accions encaminades a commutar el port, i en la resta de casos (sobre el mapa ja que només hi ha dos disparadors), a ubicar un element.

4.12 Funció changeState(et)

Tal i com s'aprecia en l'apartat anterior, en el cas de que qui hagi disparat la funció tingui assignada la classe `onswitch_o offswitch_o`, és a dir, que es tracti de un DO, la funció haurà d'implementar el codi necessari per tal de commutar aquest port i òbviament enviar l'ordre al dispositiu. Per tal de que intuïtivament per a l'usuari quedi reflectit en el mapa primerament caldrà executar la funció `changeState`. Aquesta funció li passarem l'element que hem polsat i que per tant, ha disparat la funció, i s'encarregarà de commutar la seva classe per tal de que es percebi una commutació a nivell visual per l'usuari. Tan simple com modificar la classe de l'element:



```
function changeState(et){

    switch (et.className){
case "onswitch_o": et.className = "offswitch_o"; break;
case "offswitch_o": et.className = "onswitch_o"; break;
case "onswitch_i": et.className = "offswitch_i"; break;
case "offswitch_i": et.className = "onswitch_i"; break;
    }

}
```

Com es pot deduir del codi, aquesta funció també ens servirà per commutar l'estat de sensors digitals i mostrar imatges diferents si el seu estat és `ON` o `OFF`.

Un cop modificat l'estat es crida a la funció refresh().

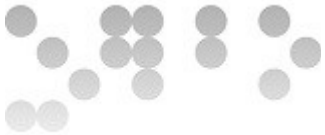
4.13 Funció refresh i compara(n,t,v)

Aquestes funcions són bàsiques per al funcionament de tota l'aplicació i seran recurrentment accedides pel codi durant l'execució de l'aplicació.

El que fan és comparar el que mostra l'aplicació i el que es desprèn de la lectura dels ports del dispositiu. En cas de trobar diferències, implementen accions per tal de que hi hagi coherència entre el que mostra l'aplicació i el que tenim realment o almenys el que esta llegint el dispositiu i ens retorna en forma de lectura de ports.

A nivell operatiu `refresh()` és molt similar a `createMenu(tx)`. Ja que en essència el que fa és fer una petició de lectura de ports del dispositiu i processar el que aquest ens retorna. La diferència radica en el que fa amb les dades que n'extreu:

```
function refresh(){
```



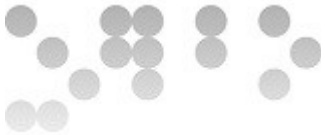
```
var r = REQ("lsc+");

r.onreadystatechange = function(){
     if (r.readyState == 4 && r.status == 200){

                    var tx = r.responseText;
                    var a = tx.split("\n");

                    for (var i=0; i<a.length; i++) {
                    b=a[i].split(":");
                    if (a[i] != "DONE")

                        try {compara(b[0],b[1].substring(0,2),b[2]);}
catch(e){}
    } }
         else {
              log(r.readyState+" "+r.statusText+" \n");
        }}}
```

Tal i com indica el codi, envia les dades a la funció `compara(n,t,v)`, que tenint les dades de cada port valora que cal fer amb cadascun d'ells.

Compara primerament busca si existeix l'objecte amb el nom del port, és a dir, si aquest port ha estat situat sobre el mapa.

En cas afirmatiu n'extrau la classe per saber de quin tipus de port es tracta i a partir d'això implementa l'acció a dur a treme en funció del tipus de port.

```
function compara (nom, tipo, valor){

    if (ID(nom)){
    var n = ID(nom);
    var c = n.className;
```



```
switch (c){
```

Tal i com succeïa a l'hora de inserir un port al mapa, el ports estan dividits en 4 classes: sortida digital, sortida analògica, entrada digital i entrada analògica.

Tal i com esta implementat el codi mentre el l'aplicació estigui connectada al dispositiu sempre enviarà ordres per tal de que el que tingui als ports de sortida sigui copiat al dispositiu i a l'inrevés. Per entendre el que volem dir amb això, analitzem el funcionament del codi:

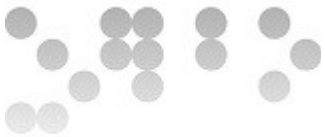
```
//Valores de salida
        case "onswitch_o": if (valor != "ON"){ z =
REQ("set+"+nom+"+ON"); log(z.responseText); } break;
                case "offswitch_o": if (valor != "OFF"){ z =
REQ("set+"+nom+"+OFF"); log(z.responseText); } break;
```

En aquest supòsit compara els ports de sortida digital, el que vol dir el valor de l'interruptor que tenim dibuixat sobre l'aplicació preval sobre el valor de la lectura. Si la classe es `onswitch_o` corresponent a un interruptor obert i el valor és diferent a `ON`, envia la petició de modificació del valor del port a `ON`.

Si la classe es `onswitch_off` corresponent a un interruptor tancat i el valor és diferent a `OFF`, envia la petició de modificació del valor del port a `OFF`.

En el cas de ports d'entrada l'operació és a la inversa. El valor del port preval sobre el que tenim a l'aplicació, aleshores:

```
//valores de entrada
case "onswitch_i": if (valor != "ON") {changeState(n);} break;
case "offswitch_i": if (valor != "OFF") {changeState(n);} break;
```



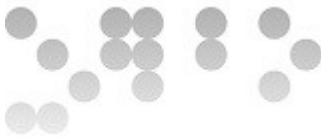
Es modifica la classe de l'interruptor perquè passi a mostrar la classe que es correspon amb l'estat del port a través de la funció `changeState` que únicament conmuta entre els dos estats modificant la classe `onswitch_i` i `offswitch_i`.

Les funcions per a ports analògics fan el mateix, si fa no fa, però tenint en compte que el format de les dades es diferent:

```
//Valores de salida analog
case "analogswitch_o": if (valor != ID(nom).value) {setValue( nom,
ID(nom).value );}  break;

//Valores de entrada analog
case "analogswitch_i": if (valor != ID(nom).value) {showValue( nom,
valor );}  break;
```

En aquest cas criden a unes funcions que s'encarreguen de fer el tractament de les dades. Modificant el valor del port o mostrant el valor llegit en cada cas.

```
function setValue(n,v){

   z = REQ("set+"+n+"+"+v);
   log(z.responseText);

}

function showValue(n,v){
    ID(n).innerHTML(v);
}
```



4.14 Funció placeButton

Tal i com es comentava a l'apartat `tapandhold(e)`, es podia donar el cas de que aquesta funció (`tapandhold`) fòs disparada pel mapa i per tant, l'acció duta a terme per l'aplicació hauria de ser ubicar l'element que haguem prèviament seleccionat a través del menú lateral.

Per fer-ho, s'implementa la funció `placeButton()`. Aquesta funció ha de llegir correctament la posició del dit sobre la pantalla i ubicar allà l'element.

Per saber la posició cal tenir en compte diversos factors:

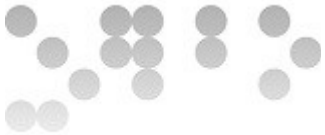
- Caldrà afegir l'element al llistat d'elements esborrables per que pugui ésser posteriorment esborrat.
- La posició on s'ha d'ubicar ha de ser en relació amb el contenidor del mapa (`scroller`), ja que el mapa es pot moure a través de la pantalla i es pot fer zoom i l'element ha de continuar encastat al mapa després de dur a terme qualsevol d'aquest tipus d'accions.

Dit això, el primer que s'implementa és el codi necessari per clonar l'element `option` que tenim seleccionat al menú *d'afegir element* i copiat al menú *d'eliminar element* i automàticament deshabilitar l'opció al primer menú dels comentats.

Això vol dir:

```
function placeButton(){


    var no = d.options[d.selectedIndex].cloneNode(true);
    d.options[d.selectedIndex].disabled = true;

    AC(r,no);
```

A continuació es determina la posició de l'objecte en qüestió, que serà el mateix que hagi disparat la funció `addbutton()`:



```
object.style.left = coordenadaX+'px';
        object.style.top = coordenadaY+'px';
        object.style.display = 'block';

//Se cancelan los disparadores de eventos
imagen.removeEventListener('touchstart', tapandhold);
imagen.removeEventListener('touchstart', getMousePosition);
imagen.removeEventListener('touchend', holding);
```

Finalment un cop situat l'objecte cal cancel·lar els disparadors d'events que estan relacionats amb el contenidor del mapa (`scroller`).

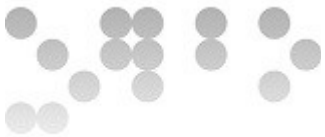
4.15 Funció getMousePosition(e)

Una qüestió important que no ha estat encara explicada és com les variables `coordenadaX` i `coordenadaY` obtenen el valor exacte de la posició no s'ha polsat sobre el mapa. És aquí on la funció `getMousePosition` fa la seva tasca.

Tal i com es descriu en apartats anteriors, quan l'usuari selecciona un element i l'afegeix, en un moment del codi s'activen uns disparadors sobre el contenidor del mapa:

```
imagen.addEventListener('touchstart', tapandhold, false);
imagen.addEventListener('touchstart', getMousePosition, false);
imagen.addEventListener('touchend',
holding = function(){ hold = false; clearTimeout(cuenta); }, false);
```

Si recordem el que es comenta en el seu corresponent apartat, el primer disparador comprova que la manera en que interactuem amb el display del l'aparell es la correcta (un sol dit), i dispara una funció que comprova que després d'un temps 'x' que pot



variar, si la variable `hold` , que denota que encara estem apretant, encara és `true`.

És important veure que el tercer disparador avalua a `false` aquesta variable quan deixem d'apretar amb el dit i a més cancel·la el disparador de la funció `setTimeout` anomenat `cuenta`. És a dir, que si abans dels 'x' segons deixem d'apretar, es cancel·larà la funció `tapandhold`.

Tornant a la part que ens ocupa en aquest apartat: Si ens fixem entre aquests dos disparadors comentats, hi ha un altre que dispara la funció `getMousePosition` quan comencem a prémer justament després d'activar `tapandhold`.

Aquesta funció el que fa es capturar la posició del dit sobre el display en relació amb `scroller`, és a dir, el contenidor de la imatge del mapa que haguem carregat.

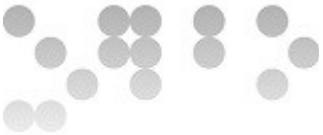
```
function getMousePosition(e){
        coordenadaX = (-myScroll.x)+(e.layerX - pageXOffset);
        coordenadaY = (-myScroll.y)+(e.layerY - pageYOffset);;
   }
```

La manera de fer el càlcul és la següent:

Primerament agafa la variable `myscroll` que ens indica el desplaçament de la capa `scroller` en abscisses i ordenades (ja sigui x o y) respecte a la capa contenidora `wrapper`. Es calcula desde el vèrtex de la part superior esquerre situant el punt inicial en (0,0). Per tant tots els moviments seran en negatiu. Com nosaltres haurem de fer el camí invers caldrà negar aquest valor. És a dir, si la capa es mou dos punts a l'esquerra -2 nosaltres ens haurem de moure +2 per trobar el punt que volem.

En segon lloc una vegada hem neutralitzat el moviment de la capa `scroller` i tenint en compte que qui haurà disparat l'event és aquesta mateixa capa extraurem el seu punt d'abscisses o ordenades i restarem el possible *scrolling* que pugui haver. Com a resultat final obtenim la posició correcte en cada moment.

Cal remarcar molt que en la implementació d'aquesta funció es fan servir variables



pròpies de l'objecte `iScroll` com ara `iScroll.x` o `iScrolly`, junt amb altres variables que no son estàndard com `layerX`, `layerY` o `pageXOffset`, `pageYOffset` i per tant, algunes famílies de navegadors no funcionin correctament.

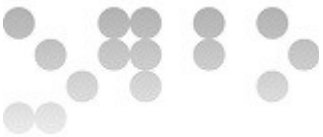
4.16 Funció removebutton

Ja hem descrit com funciona el mecanisme per afegir elements al mapa. A continuació descriurem com treure'n i per fer-ho ens servirem de la funció `removebutton()`.

Aquesta funció és cridada quan seleccionem un element `option` al llistat d'elements a esborrar i premem el botó *remove button.*

```
function removebutton(){

        if (r.selectedIndex > -1){
        var nom = r.options[r.selectedIndex].text;
        ID(nom).style.display = 'none';
        r.removeChild(r.options[r.selectedIndex]);

        for (var i = 0; i < d.options.length; i++){
            if( d.options[i].text == nom) {d.options[i].disabled =
false;}}


        menu('derecha');
        log("Removed!");
        }
         else { menu('derecha'); alert ("Select a valid value!");}

    }
```

El primer que fa és comprovar que hi ha algun element seleccionat. Seguidament extrau el nom de l'element que volem eliminar i el busca per la seva ID, que serà



l'objecte que el representa sobre el mapa i aleshores en realitat el que fa és ocultar-lo amb un `display = none` i eliminar l'opció del menú.

En cas de que vulgui tornar a afegir aquest element només caldrà fer un `display = block` i posicionar-lo.

Finalment el bucle `for` fa un escombrat de tot el menú *afegir* per tornar a activar l'opció de afegir aquest element de nou. En cas de no seleccionar un element vàlid, mostra un missatge d'error.

4.17 Diverses plantes

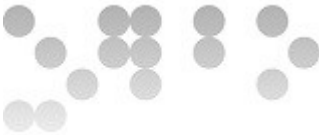
Ja podem identificar-nos, inserir un plànol i ubicar ports que es comuniquen amb el dispositiu però que passa si tenim un habitacle amb diverses plantes?

Per tal de resoldre aquesta qüestió, l'estratègia a seguir a nivell tècnic és molt similar a la seguida per tal d'implementar les finestres lliscants dels menús, és a dir, per a cada mapa afegit s'autogenerarà un contenidor que contindrà un *wrapper* i un *scroller*. Aquest *scroller* contindrà alhora la imatge afegida. Això suposa una sèrie de replantejaments i modificacions respecte al codi implementat fins al moment.

Per començar caldrà implementar el botó al menú que cridarà la funció que implementarà la nova funcionalitat:

```
<tr>
    <th colspan="3">Add maps</th>
</tr>

<tr>
    <td colspan="3">
    <input type="button" value="Search..." onclick="carrete()"> </p>
    </td>
</tr>
```

Si ens fixem, aquest botó crida a la antiga funció *carrete(),* ja existent, i que carregava

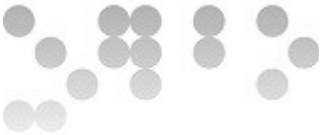


el primer mapa a la pantalla de *login.* Perquè la mateixa funció es pogués reutilitzar optimitzant el codi s'han hagut d'implementar una sèrie de modificacions:

```
//Cargar imágenes
function carrete()
{

 navigator.camera.getPicture(onSuccess, onFail, { quality:
90,destinationType: destinationType.FILE_URI, sourceType:
Camera.PictureSourceType.PHOTOLIBRARY });


}

function onSuccess(imageURI) {

if (!(ID("mapas"))){

        cuerpo.innerHTML='<div id="mapas"> </div>';
        logar();

 }
```

Si ens fixem en el codi veurem que el primer que fa, un cop carregada la imatge amb èxit, és comprovar si aquesta funció ha estat cridada desde el *login* o desde el botó d'afegir mapa, ja que si aquesta crida es fa per primera vegada desde el *login*, haurà de substituir el primer menú pel mapa i cridar la funció `login`, mentre que si la crida es fa amb la finalitat d'afegir mapa, el que vol dir que ja hi ha com a mínim un mapa carregat, s'haurà/an de mantindre i afegir-ne un de nou.

La manera de saber-ho és saber si s'ha creat la caixa amb la `id=mapas`. Si no està creada, voldrà dir que es la primera vegada que es crida la funció desde el menú de

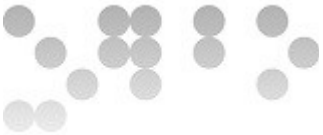


login, i per tant, l'haurem de crear. En canvi si ja existeix voldrà dir que ja hi ha algun mapa carregat.

Cal tenir en compte també que dins el `div mapas` s'hauran d'implementar imatges amb propietats especials per poder navegar sobre els mapes i poder fer zoom. Si recordem la manera com fèiem servir `iScroll`, veurem que una vegada creada l'estructura `wrapper + scroller`, calia instanciar un objecte `iScroll` que és el que donava les propietats especials. Que passa doncs si volem tenir diversos elements d'aquest tipus?

Primerament caldrà instanciar un per cada finestra navegable i amb zoom que vulguem. No ens servirà crear diversos elements amb `id=wrapper i id=scroller` i després instanciar un `iScroll` per a totes les finestres. Així que el primer que caldrà fer es referenciar l'estil dels elements `wrapper` i `scroller` com a classes:

```
/* Estilos para iscroll-zoom*/

.wrapper {
    position: inherit;
    z-index: 1;
    top: 0px;
    bottom: 0px;
    left: 0px;
    width: 100%;
    height: 100%;
    background: #ccc;
    overflow: hidden;
}

.scroller {
    position: absolute;
    z-index: 3;
    -webkit-tap-highlight-color: rgba(0,0,0,0);
```

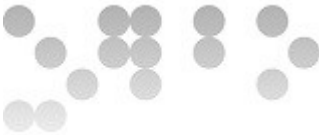


```
    width: 1200px;
    -webkit-transform: translateZ(0);
    -moz-transform: translateZ(0);
    -ms-transform: translateZ(0);
    -o-transform: translateZ(0);
    transform: translateZ(0);
    -webkit-touch-callout: none;
    -webkit-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
    -webkit-text-size-adjust: none;
    -moz-text-size-adjust: none;
    -ms-text-size-adjust: none;
    -o-text-size-adjust: none;
    text-size-adjust: none;
    background: #fff;
}
```

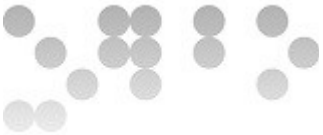
Un cop fet això cada vegada que s'afegeixi un nou mapa es crea la instancia `iScroll` que tindrà aquest mateix nom seguit d'un nombre que atorgarà la veriable `mapnumber`. Aquesta és una variable global de tipus enter que serà l'encarregada de comptabilitzar el nombre de mapes afegits fins al moment. Cada vegada que s'afegeixi un mapa a través de la funció `addMap`, aquesta variable incrementarà el seu valor i ens servirà per donar nom a diversos elements de la mateixa classe i poder distingir-los entre ells.

Al mateix temps es farà servir per referenciar les caixes `wrapper` i `scroller`.

```
// Llama a la función que añade una planta
addMap(imageURI);


//Aplicamos la librería iscroll-zoom sobre el código HTML creado
```



```
        window["myScroll"+mapnumber] = new
IScroll('#wrapper'+mapnumber, {
                                zoom: true,
                                scrollX: true,
                                scrollY: true,
                                mouseWheel: true,
                                wheelAction: 'zoom'
                                });
        document.addEventListener('touchmove', function (e)
{ e.preventDefault(); }, false);


    }

    function onFail(message) {
        log('Failed because: ' + message);
    }



function addMap(URI){

//mapnumber esta inicializado en 0 al inicio de la aplicación
    ++ mapnumber;

 //Creamos el div plano
    var a=CE('div','classplano','plano'+mapnumber);
    var b='<div id="wrapper'+mapnumber+'" class="wrapper"> <div
id="scroller'+mapnumber+'" class="scroller"><img id=miimagen
src="'+URI+'"> </div> </div>'
    a.innerHTML=b;

    //El primero es el div donde se meten los mapas y el segundo es el
```



```
select donde se insertara la opcion
    var c=ID('mapas');
    m=ID('mapas_sel');

//añadimos el div 'plano' a 'mapas'
AC(c,a);
```

Si analitzem detingudament el codi veurem que aquesta funció crea l'estructura anteriorment comentada i la encapsula dins d'un `div` anomenat `plano` i de `clase=classplano`, és a dir que el primer mapa serà:

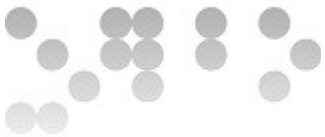
`plano1[wrapper1[scroller1(imatge1)]]`

El següent seria `plano2` i així successivament. Una vegada creada l'estructura la situa com a fill del `div mapas`.

Fins a aquest punt, les noves imatges inserides s'apilarien una sobre l'altra o potser una al costat de l'altra, sense seguir cap ordre i sense poder moure'ns entre elles. Cal doncs dissenyar un sistema que ens permeti saltar d'una a l'altra de manera que puguem seleccionar una o una altra, eliminar la que no ens sigui necessària o mostrar la que necessitem en cada moment.

Es per això que al menú crearem un desplegable de tipus `select` i dos botons que cridaran dues funcions que s'encarregaran de mostrar el mapa que vulguem.

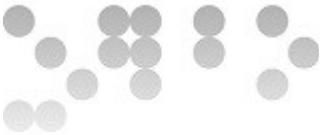
```
<tr>
    <th colspan="3">Manage maps</th>
</tr>

<tr>

 <td><select name="port" id="mapas_sel"></select> </td>

 <td><input type="button" value="Show" onclick="switchMap()"></input>
</td>
```



```
 <td><input type="button" value="Delete" onclick="delMap()"></input>
</td>

</tr>
```

Però per que puguem seleccionar alguna de les imatges inserides, primer cal anar omplint el desplegable i que només ens mostri una imatge alhora. Per tant, caldrà ampliar la funció `addMap` amb el següent codi:

```
    for(i = 0; i <mapnumber; i++){
        if(m.options[i]){ m.options[i].activo= false;}

    }


    //Creamos la opcion
    var op=CE("option");
    op.text = 'plano'+mapnumber;
    op.number = mapnumber;
    op.activo = true;



    //añadimos la opcion al select
    AC(m,op);
    m.selectedIndex=(m.length)-1;


    for(i = 1; i <=mapnumber; i++){
        var d=ID('plano'+i);
        if(d){
```



```
            d.className = 'classplano right';}
    }

    //Hacemos un display block al añadido y el resto lo ocultamos
    a.className='classplano center';
}
```

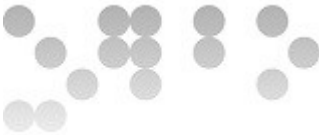
Cadascún dels elements `option` del desplegable, conté 3 atributs: `text`, `number` i `activo`. El primer és el nom que es podrà llegir al desplegable que és el nom genèric *plano* seguit del valor actual de `mapnumber`, el segon serà un enter amb el valor actual `mapnumber` i el darrer serà un booleà inicialitzat a `true`, que ens indicarà si aquesta és la imatge que s'està mostrant en aquell moment, atès que pot ser que es modifiqui el valor del desplegable sense prémer cap botó i aleshores el mapa actiu i el que marcaria el desplegable serien diferents.

En el moment de inserir una nova imatge aquesta passarà a ser la mostrada i és per això que primer hi ha un bucle `for` que avalua totes les opcions anteriorment inserides amb el valor `activo=false` i posteriorment genera la opció que per defecte inicialitza aquest valor a `true`.

Tal i com passa amb el sistema de finestres lliscants, per tal de mostrar només l'última inserida el que fa és ocultar totes les anteriors modificant la seva classe mitjançant un bucle `for` i després mostrar la que acabem de generar.

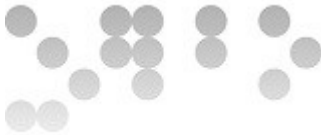
Les classes afegides per poder fer aquest efecte possible són:

```
.classplano{
    position: absolute;
    top:0px; left:0;
    width: 100%;
    height: 100%;
```



```
    -webkit-transform: translate3d(0, 0, 0);
    transform: translate3d(0, 0, 0);

}


.classplano.left {
    -webkit-transform: translate3d(-200%, 0, 0);
    transform: translate3d(-200%, 0, 0);
}

/* Estilo de capa central */

.classplano.center {
    -webkit-transform: translate3d(0, 0, 0);
    transform: translate3d(0, 0, 0);
}

/* Estilo de capa derecha */

.classplano.right {
    -webkit-transform: translate3d(200%, 0, 0);
    transform: translate3d(200%, 0, 0);
}
```

Si ens fixem el moviment fora de la pantalla és d'un *200%* i no un *100%* com succeïa amb les classes `page.totalleft i page.totalright`. Això és degut a que com la seva referència és, en definitiva, la capa `manu_principal`, si es mouen només un *100%* les imatges no mostrades, acaben sobre el menú dret o esquerre.



4.18 Funció switchMap

Aquesta funció que es cridada pel botó corresponent al menú, tal i com es descriu a l'apartat anterior, llegeix quin es el mapa seleccionat al `select mapas_sel` i el mostra a la pantalla principal. Recordem que la variable global `m` referencía el `select mapas_sel`.

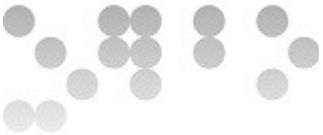
Primer comprova que hi ha algun mapa disponible (índex major que `-1`), ja que com veurem més endavant, la opció d'esborrar mapes ens permet esborrar-los tots.

La variable local `n` guarda el valor de l'atribut `number` de la opció seleccionada i a través d'aquest valor guarda l'objecte `plano` que es correspon amb aquesta opció a la variable `map`.

```
function switchMap(){
    if(m.selectedIndex > -1){
    var n = m.options[m.selectedIndex].number
    var map = ID('plano'+n);
```

Seguint el mètode de la funció `addMap`, mitjançant un bucle `for` va escombrant totes les opcions del `select`, saltant-se les que han estat eliminades, avalua el seu atribut `activo=false` i mou l'element `plano` que es correspon amb l'opció fora de la pantalla modificant la seva classe a `classplano.right`. Excepte si és l'opció seleccionada, que el manté com a `true` i el mostra.

```
//Los ocultamos todos
for (i=0; i<=mapnumber; i++){
   if(m.options[i]){m.options[i].activo= false;
   ID('plano'+(m.options[i].number)).className='classplano right';

if(m.options[i].number == n ){m.options[i].activo=true;}
        }
    }
```



```
  //Mostramos el seleccionado
    map.className='classplano center';
    }

    else {alert('Error: No map selected!');}

}
```

4.19 Funció delMap

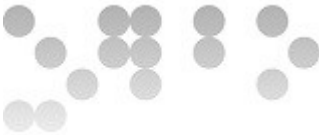
Igual com passa amb `switchMap`, cal prémer el botó *Delete* de la secció *Manage Maps,* per tal de cridar aquesta funció que tenint en compte el mapa seleccionat al `select mapas_sel`, executarà el codi que hi ha més a baix per tal d'eliminar el mapa corresponent.

La funció no esborra automàticament tots els ports incrustats sobre el mapa en qüestió, sino que caldria fer-ho manualment a través del menú si es volen tornar a situar de nou.

```
function delMap(){

    if (m.selectedIndex > -1){
    var n = m.options[m.selectedIndex].number
    var map = ID('plano'+n);
```

Aquesta primera part fa exactament el mateix que `switchMap`, comprova si hi ha mapa disponible actiu, guarda l'atribut `number` i guarda l'objecte `plano` seleccionat a `map`.

A continuació abans d'esborrar la corresponent opció al menú desplegable, mou l'element seleccionat al desplegable una posició enrere per defecte si pot, en cas contrari (si el mapa a esborrar es el primer que esta en la posició 0) ho fa endavant. Si cap de les dues és possible perquè és l'últim, no es mou i la opció queda buida



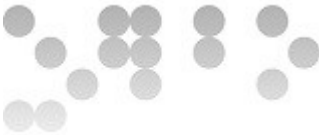
després d'esborrar.

```
if(m.options[m.selectedIndex-1]){
        m.selectedIndex=m.selectedIndex-1;
        m.remove(m.selectedIndex+1);

    }

    else{
        m.selectedIndex=m.selectedIndex+1;
        if (m.selectedIndex!=-1){m.remove(m.selectedIndex-1);}
        else{m.remove(0);}
    }

    if(m.selectedIndex > -1){
    ID('plano'+
(m.options[m.selectedIndex].number)).className='classplano center'
    m.options[m.selectedIndex].activo=true;


    }
    var p=ID('mapas');
        p.removeChild(map);}
    else {alert('Error: No map selected!');}
```

D'aquest algoritme se'n desprèn la possibilitat de que s'esborrin tots els mapes per tant es valora aquesta opció i en cas de cridar a la funció sense cap mapa seleccionat, es mostra un error amb una finestra d'alerta.

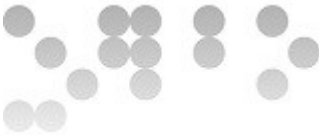
4.20 Settings

Finalment s'afegeixen al menú de preferències una sèrie de botons que criden



funcions simples per modificar opcions bàsiques.

```
<div id="settings">
   <header class="cabecera">SETTINGS</header>
   <nav id="menu">
      <table> <form name="settings_form">

     <tr>
     <td> Refresh frequency </td>
     <td><input type="range" name="frecr" id="frecr" min="1" max="60"
value="10"
                                  onchange="setRefresh(this.value)"
oninput="setRefresh(this.value)"></input> </td> </tr>

<tr> <td> Reload Device </td>
<td> <input type="button" value="Reload" onclick="setting('1')
"></input></td> </tr>

<tr> <td> Up Time </td>
<td> <input type="button" value="UPT" onclick="setting('2')
"></input></td> </tr>

<tr> <td> Clear Config</td>
<td> <input type="button" value="Clear" onclick="setting('3')
"></input></td> </tr>

<tr> <td> Default </td>
<td> <input type="button" value="Default" onclick="setting('4')
"></input></td> </tr>

              </form> </table>
       </nav>
  </div>
```



La primera funció permet modificar la freqüència de refresc que es defineix per defecte cada 10 segons. És a dir, cada 10 segons, l'aplicació fa una consulta de l'estat dels ports al dispositiu i actualitza les dades a l'aplicació o als ports, segons es requereixi.

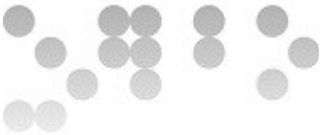
```
function  setRefresh(freq){

    freq = freq * 1000;
    clearInterval(freqRefresco);
    freqRefresco = setInterval(refresh, freq);

}
```

Per definir un valor per defecte, es defineix el valor `freqRefresco` en el moment de creació del menú de l'aplicació al final de la funció `createMenu(tx)` afegint les següents línies de codi:

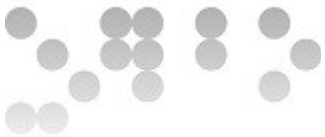
```
   freqRefresco = setInterval(refresh, 10000); //Cada 10s
```

Pel que fa a la resta de botons, criden totes a la mateixa funció però passant un valor:

```
function setting(v){
            var w,r;

            switch(v){

        case 1: w ="rel+"; break;
        case 2: w ="upt+"; break;
        case 3: w ="clr+"; break;
        case 4: w ="def+"; break;
```



```
    }

  r = REQ(w);
   r.onreadystatechange = function(){
            menu('izquierda');
            log(r.readyState+" "+r.statusText+" \n");
        }


}
```



5 Conclusions

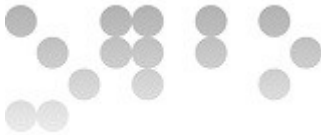
Arribats a aquest punt, cal avaluar les fites plantejades inicialment i quina ha estat la seva consecució. D'aquesta manera podem quantificar d'alguna manera si hem aconseguit assolir els objectius marcats inicialment i si hi ha algun aspecte a millorar.

5.1 Diagrama final

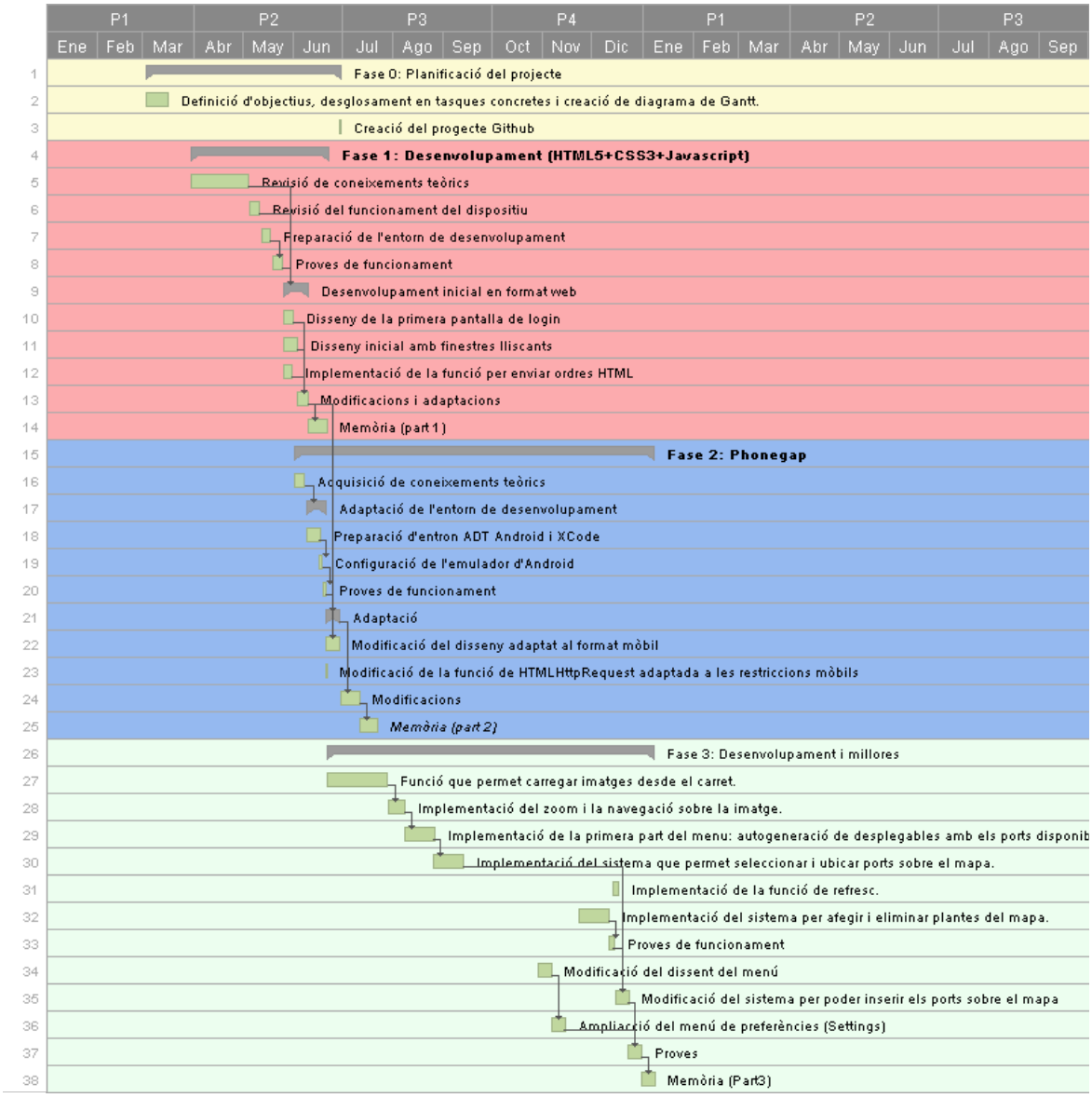
Si recordem, al capítol 2 es definien uns objectius en base als quals es desenvoluparia la planificació inicial del projecte desglossant aquests objectius en tasques i subtasques i assignant a aquestes tasques un espai i una previsió temporal.

A mida que el projecte ha anat avançant, aquestes previsions s'han anat modificant per adaptar-se als requeriments del projecte en cada moment.

Comprovem quines son finalment aquestes tasques i com queden distribuïdes en el temps:

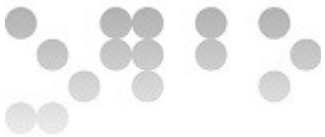


Tasca	Data inici	Data fi	Duració	Coneixements aplicats
Fase 0: Planificació del projecte	03/03/14	29/06/14	102	
Definició d'objectius, desglosament en tasques	03/03/14	15/03/14	12	IMPLANTACIÓ DE SISTEMES DE PROGRAMARI LLIURE.
Creació del progecte Github	29/06/14	29/06/14	1	INTRODUCCIÓ AL DESENVOLUPAMENT DE PROGRAMARI.
Fase 1: Desenvolupament	**30/03/14**	**21/06/14**	**73**	
Revisió de coneixements teòrics	30/03/14	03/05/14	31	DESENVOLUPAMENT D'APLICACIONS WEB, XARXES OBERTES, BASES DE DADES.
Revisió del funcionament del dispositiu	05/05/14	10/05/14	6	NOUS CONEIXEMENTS.
Preparació de l'entorn de desenvolupament	13/05/14	17/05/14	5	INTRODUCCIÓ AL DESENVOLUPAMENT DE PROGRAMARI.
Proves de funcionament	19/05/14	24/05/14	6	INTRODUCCIÓ AL DESENVOLUPAMENT DE PROGRAMARI.
Desenvolupament inicial en format web	26/05/14	09/06/14	13	
Disseny de la primera pantalla de login	26/05/14	31/05/14	6	DESENVOLUPAMENT D'APLICACIONS WEB.
Disseny inicial amb finestres lliscants	26/05/14	02/06/14	7	DESENVOLUPAMENT D'APLICACIONS WEB.
Implementació de la funció per enviar ordres	26/05/14	30/05/14	5	DESENVOLUPAMENT D'APLICACIONS WEB.
Modificacions i adaptacions	03/06/14	09/06/14	6	DESENVOLUPAMENT D'APLICACIONS WEB.
Memòria (part 1)	10/06/14	21/06/14	11	TREBALL FINAL DE MÁSTER.
Fase 2: Phonegap	**02/06/14**	**07/01/15**	**189**	
Adquisició de coneixements teòrics	02/06/14	07/06/14	6	NOUS CONEIXEMENTS.
Adaptació de l'entorn de desenvolupament	09/06/14	20/06/14	11	
Preparació d'entron ADT Android i XCode	09/06/14	16/06/14	7	NOUS CONEIXEMENTS, DESENVOLUPAMENT D'APLICACIONS WEB.
Configuració de l'emulador d'Android	17/06/14	18/06/14	2	NOUS CONEIXEMENTS, DESENVOLUPAMENT D'APLICACIONS WEB.
Proves de funcionament	19/06/14	20/06/14	2	INTRODUCCIÓ AL DESENVOLUPAMENT DE PROGRAMARI.
Adaptació	21/06/14	28/06/14	7	
Modificació del disseny adaptat al format mòbil	21/06/14	28/06/14	7	NOUS CONEIXEMENTS, DESENVOLUPAMENT D'APLICACIONS WEB.
Modificació de la funció de HTMLHttpRequest	21/06/14	21/06/14	1	NOUS CONEIXEMENTS, DESENVOLUPAMENT D'APLICACIONS WEB.
Modificacions	30/06/14	11/07/14	11	
Memòria (part 2)	12/07/14	22/07/14	9	TREBALL FINAL DE MÁSTER
Fase 3: Desenvolupament i millores	**22/06/14**	**07/01/15**	**172**	
Funció que permet carregar imatges desde el	22/06/14	28/07/14	32	NOUS CONEIXEMENTS, DESENVOLUPAMENT D'APLICACIONS WEB.
Implementació del zoom i la navegació sobre la	29/07/14	07/08/14	9	NOUS CONEIXEMENTS, DESENVOLUPAMENT D'APLICACIONS WEB.
Implementació de la primera part del menu:	08/08/14	25/08/14	15	NOUS CONEIXEMENTS, DESENVOLUPAMENT D'APLICACIONS WEB.
Implementació del sistema que permet	26/08/14	12/09/14	16	NOUS CONEIXEMENTS, DESENVOLUPAMENT D'APLICACIONS WEB.
Implementació de la funció de refresc.	14/12/14	16/12/14	3	NOUS CONEIXEMENTS, DESENVOLUPAMENT D'APLICACIONS WEB.
Implementació del sistema per afegir i eliminar	23/11/14	10/12/14	16	NOUS CONEIXEMENTS, DESENVOLUPAMENT D'APLICACIONS WEB.
Proves de funcionament	11/12/14	13/12/14	3	NOUS CONEIXEMENTS, DESENVOLUPAMENT D'APLICACIONS WEB.
Modificació del dissent del menú	29/10/14	05/11/14	7	NOUS CONEIXEMENTS, DESENVOLUPAMENT D'APLICACIONS WEB.
Modificació del sistema per poder inserir els ports	15/12/14	22/12/14	7	NOUS CONEIXEMENTS, DESENVOLUPAMENT D'APLICACIONS WEB.
Ampliacció del menú de preferències (Settings)	06/11/14	13/11/14	7	NOUS CONEIXEMENTS, DESENVOLUPAMENT D'APLICACIONS WEB.
Proves	23/12/14	30/12/14	7	INTRODUCCIÓ AL DESENVOLUPAMENT DE PROGRAMARI.
Memòria (Part3)	31/12/14	06/01/15	7	TREBALL FINAL DE MÁSTER



5.2 Coneixements aplicats i adquirits

Si ens fixem en la columna *Coneixements aplicats*, podem veure en quina assignatura del M*àster en Programari Lliure* es van adquirir els coneixements necessaris per desenvolupar cadascuna de les tasques.



El projecte ha quedat finalment estructurat en 4 fases: una primera fase de planificació, 2 fases de desenvolupament i una fase d'adaptació entre les dues anteriors.

A la **Fase 0,** que ha consistit en la planificació inicial, els aspectes relacionats amb la gestió de projectes com la *gestió de l'abast (*en la definició i avaluació dels objectius del projecte) o la *gestió del temps (*en la implementació de la planificació: diagrama de Gantt), han estat aplicats per tal d'administrar i estructurar degudament i de manera ordenada el projecte al llarg del temps, optimitzant així la seva consecució tenint en compte el context: temps disponible limitat, medi de comunicació telemàtic, etc.

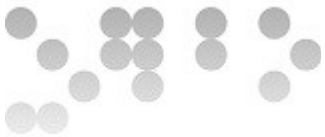
En aquesta primera fase s'han aplicat també els coneixements de Sistemes de control de versions adquirits a *Introducció al desenvolupament de programari* per generar un repositori a Github i gestionar els seu contingut.

A la **Fase 1**, que és la primera fase de desenvolupament, els coneixements d'*Introducció al desenvolupament de programari* concretament l'apartat relacionat amb *Entorns de desenvolupament,* és el que ha fet possible que ja ens fossin familiars els entorns de desenvolupament (IDE) i sabéssim treballar amb ells, sobretot amb l'ADT d'Android que no és més que una adaptació d'Eclipse optimitzada per al desenvolupament d'aplicacions mòbils per a aquest sistema.

Pel que fa a l'implementació de l'aplicació hem aplicat els coneixements teòrics i pràctics de *Desenvolupament d'aplicacions web* (*HTTP, HTML, CSS i Javascript) que* ens han servit de base per desenvolupar un primer disseny a partir del qual començar a treballar.

Ampliant aquests coneixements, hem après a implementar webs dinàmiques i visualment atractives i modernes que interactuessin amb l'usuari modificant continguts i implementant efectes visuals a partir de la integració de les darreres versions de les tecnologies web anteriorment esmentades: HTML5, CSS3 i Javascript.

Per tal de que el primer disseny es pogués comunicar amb el dispositiu i mostrar dinàmicament els resultats hem hagut d'aprendre a implementar la tècnica de programació *AJAX* a través de l'API *XMLHttpRequest* i entendre com funciona la



comunicació asíncrona per obtenir dades a través del protocol HTTP i com aquestes dades modifiquen el *DOM* sense necessitat de recarregar tot el contingut.

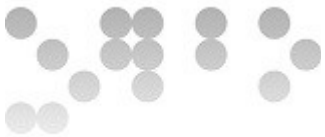
L a **Fase 2** ha estat una fase eminentment d'aprenentatge i familiarització amb Phonegap. Hem après com gràcies a aquesta eina es pot optimitzar el desenvolupament d'aplicacions per a dispositius mòbils a partir de tecnologies fins ara utilitzades per al desenvolupament web, de manera que generant un disseny d'aplicació màster, aquest es pot adaptar pràcticament a qualsevol plataforma mòbil a partir de unes simples comandes.

Al mateix temps també hem entès que fer servir una tecnologia com aquesta implica unes contrapartides importants com poden ser deficiències o limitacions en les eines de desenvolupament i depuració, ja que no es tracta d'un llenguatge de programació en si mateix sino d'un pont de comunicació o una traducció entre el llenguatge nadiu del sistema i els llenguatges web. Això significa que al ser híbrid, cadascun dels sistemes s'ha de depurar per separat: si es depura com una aplicació web, la part nadiva no es pot depurar i a la inversa. Aquest fet ha suposat una gran limitació però al mateix temps ha estat una gran font l'adquisició d'experiència que ens ha empès a entendre el funcionament de les eines de depuració i del propi Phonegap.

En el camí d'adaptació del format web a format mòbil (amb Phonegap) hem descobert també algunes tècniques utilitzades amb freqüència entre la comunitat de desenvolupadors d'aquest tipus d'aplicacions fruit de l'evolució global en la manera de dissenyar i programar i l'intercanvi de coneixements via internet: fòrums, blogs, pàgines web, etc.

Alguns exemples d'això són l'ús *sprites,* les tècniques de mostrar i ocultar enlloc de instanciar o destruir objectes, ocultació de pantalles, entre d'altres.

Finalment la **Fase 3**, ha estat la fase d'adquisició d'experiència. Aquesta fase ha estat una fase d'ampliació de coneixements de manera autodidacta i de mètode prova-error. Al llarg d'aquesta fase hem après a fer nostre els coneixements en base a la pròpia



experiència i a saber com fer-los servir per superar els reptes que s'han anat presentant per assolir les fites inicialment proposades.

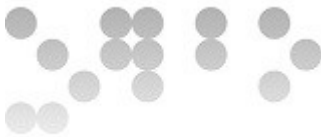
5.3 Avaluació d'objectius

Podem estructurar els objectius marcats inicialment de la següent manera:

3. Objectius a nivell funcional: Desenvolupament d'una aplicació per a dispositius mòbils per al control remot del dispositiu ODControl.
    a) Intuïtiva
    b) Funció zoom i navegació sobre els plànols
    c) Diverses plantes
    d) Pantalla de *loading*
4. Objectius a nivell personal
    1. Resolució de problemes aplicant coneixements
    2. Adquisició d'experiència
    3. Auto-aprenentatge

Podem dir que l'objectiu principal s'ha assolit sobradament: El resultat final és un aplicació gràficament agradable i dinàmica que es pot comunicar amb el dispositiu via internet per enviar i rebre ordres i mostrar-les d'una manera gràficament amigable i entenedora.

El fet de que es tracti a nivell gràfic d'una aplicació que emula altres models molt populars d'aplicacions mòbils com ara Facebook en la seva versió mòbil, ja és un símptoma de que per norma general els seus usuaris sabran en gran mesura fer-la funcionar sense cap mena de manual llevat del necessari coneixement del funcionament de l'aparell al que controla (ODControl). A més per facilitar-ne

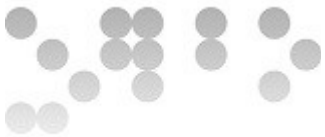


l'enteniment s'han implementat missatges d'alerta i instruccions que acompanyen a l'usuari en la seva experiència per tal de guiar-lo en el seu ús. Altres detalls com el zoom, el desplaçament del mapa o el posicionament de ports de manera tàctil també han estat elements a tenir en compte en aquest aspecte. Per tant, podríem avaluar la vessant intuïtiva del resultat final com a assolida completament i de manera satisfactòria.

Tal i com es comenta en el paràgraf anterior s'ha implementat un sistema tàctil de navegació i zoom sobre els plànols introduïts per l'usuari a partir de l'API proporcionada per la llibreria iScroll, tal i com es requeria. Aquesta API ens permet instanciar uns objectes configurables generats a partir de dos contenidors div, que interactuen amb els events tàctils creant un efecte de navegació i zoom fent servir internament propietats CSS com *transform* per modificar les seves propietats. Podem dir doncs que aquest requeriment s'ha assolit també amb èxit.

Pel que fa al problema de mostrar i controlar habitacles amb diverses plantes, s'ha implementat l'opció d'afegir i eliminar tantes plantes com es vulgui a través d'un sistema de generació automàtica. En seleccionar una imatge del carret ens crea un nou objecte contenidor que ens mostra la nova imatge mantenint la propietat lliscant i el zoom. A mida que es van afegint imatges aquest sistema instancia nous objectes amb cadascuna de les imatges i a través de CSS ens mostra el que volem en cada moment. Això fa que es perdin algunes propietats de referències en el posicionament i que posicionar alguns elements sigui una tasca una mica més complexa. Per fer-ho s'han hagut de modificar diversos algoritmes ja implementats obligant a replantejar algunes implementacions com ara el sistema de posicionament dels objectes sobre el mapa. Aquest es basava en propietats que per la pròpia naturalesa mòbil dels elements autogenerats, que es mostren o no en el display en cada moment, han desaparegut. Impedint així fer-les servir per als càlculs de posició.

Aquesta ha estat una part força enriquidora i que ens ha aportat molta experiència pel



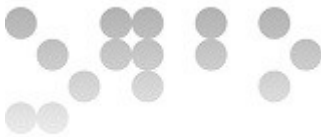
que fa a entendre la dificultat que suposa que de vegades una cosa tan simple com aquesta pugui obligar a replantejar-ho tot de nou.

Aquesta tècnica de treballar amb objectes que estan fora o dintre del display del dispositiu, que es mostren o no, modificant el seu posicionament a través estils definits en classes que es modifiquen de manera dinàmica ha estat una tècnica força utilitzada així com la de mostrar o no objectes a través de la propietat *display*. També en el cas de la pantalla de càrrega s'ha fet servir aquesta tècnica a través de la lectura d'events retornats asícronament per funcions d'objectes com els de l'*API* de l'objecte XMLHttpRequest. Per saber quan s'han rebut o no dades i es pot retirar aquesta pantalla d'espera. En aquest cas el resultat també ha estat força bo.

Per tant pel que fa al resultat s'han assolit tots els objectius marcats inicialment tot i que durant el transcurs del projecte s'han anat descobrint noves fites a desenvolupar que no s'havien valorat inicialment que es comenten posteriorment en l'apartat dedicat a millores i treballs futurs.

D'altra banda si valorem els objectius personals podem dir que per a la consecució de els objectius marcats inicialment no només s'han hagut de aplicar coneixements adquirits al llarg del màster sino que de fet, aquests només han estat una base per començar a ampliar l'infinit abast del camp que es tracta: el disseny i implementació d'aplicacions web i aplicacions mòbils basades en aquesta classe de tecnologies.

Al llarg del projecte hem après a treballar i a aprendre en comunitat a través de la xarxa, llegint informació i interactuant amb altres membres de manera col·laborativa a través per exemple de fòrums, un dels elements defensats per la filosofia del software lliure. Hem entès que aquesta es la manera de créixer i aprendre de manera eficient i exponencial: compartint i col·laborant. Ha estat aquest medi el que ens ha obert les portes a noves tècniques i maneres de treballar. Això ens ha modificat la manera de plantejar els problemes i la manera de treballar. De fet es podria dir que ha estat la part



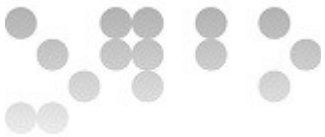
més enriquidora i important ja que ens obre la porta a una infinitat de possibilitats i de coneixement.

En base a això hem après a esprémer el màxim les possibilitats que ens aporten els entorns integrats a nivell de disseny, desenvolupament i depuració de codi, amb un llarg camí encara per recórrer, els mètodes estàndard de disseny i implementació en dispositius mòbils, tècniques per a l'eficiència del codi a nivell qualitatiu i quantitatiu i altres aspectes com la publicació d'aquestes aplicacions.

5.4 Aspectes a millorar

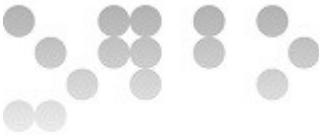
Tot i que s'han assolit els objectius marcats en l'inici del projecte, hi ha un gran marge de millora pel que fa a l'aplicació i alguns aspectes del projecte. En aquest apartat tractarem d'indicar quins aspectes es podrien haver millorat.

- **Eficiència del codi:** Tot i que s'ha intentat implementar un codi eficient en termes de funcionament, intentant que fos modular i precís aquest es un aspecte que sempre es pot millorar. A mida que l'aplicació creix sempre es poden fusionar funcions per reduir la mida del codi i no tenir duplicitats, sempre es poden modificar mètodes, etc.
- **Reducció del pes:** Deixant de banda llibreries com XUI o Jquery que ens simplifiquen força el codi i que no han estat utilitzades en aquest projecte. Sempre es poden simplificar i reduir noms de variables i funcions, eliminar observacions i reduir espais per tal d'obtenir una aplicació el més lleugera possible. El resultat és normalment una pila de caràcters alfanumèrics il·legibles però que eliminen tots aquests elements innecessaris per al funcionament que hi són per la inelegibilitat del codi.
- **Responsive design (Disseny Adaptatiu):** Aquest és un aspecte que no es va marcar inicialment i en el que l'aplicació té molt a millorar. Si ens fixem al llarg de les imatges del document, les simulacions de l'aplicació han estat fetes sobre un dispositiu mòbil standard, generalment un *iPhone4* però podria ser



que aquesta aplicació es llancés desde una tauleta o qualsevol altre dispositiu i tot i que el disseny esta implementat en base a proporcions el resultat podria ser molt diferent.

- **Recuperació de sessions:** Aquest és el més important ja que resta utilitat a l'aplicació. Una de les absències notables de l'aplicació és la capacitat de salvar una sessió ja iniciada per recuperar el seu estat. És un element a millorar molt important.



6 Bibliografia i fonts

- [wk1] Wikipedia(2012).*Historia de l'ordinador personal.* Recuperat 9 de desembre de 2014, desde http://ca.wikipedia.org/wiki/Hist%C3%B2ria_de_l%27ordinador_personal.

- [Wk2] Wikipedia(2014). *Raspberry Pi*. Recuperat el 11 de desembre de 2014, desde http://ca.wikipedia.org/wiki/Raspberry_Pi

- [wk3] Wikipedia(2014).*Hyper Text Markup Language*. Recuperat el 16 de desembre de 2014, desde http://ca.wikipedia.org/wiki/Hyper_Text_Markup_Language

- [wk4] Viquipèdia(2009). Domòtica. Recuperat el 3 de desembre de 2014, desde http://ca.wikipedia.org/wiki/Dom%C3%B2tica

- Mateu, Carles; Mas, Jordi (coord.); Megías Jiménez, David(coord)(2006). *Desenvolupament d'aplicacions web*.Barcelona: UOC.

- Gauchat, Juan Diego(2012).*El Gran Libro de HTML5, CSS3 i Javascript.* Barcelona: MARCOMBO.

- Monsur Hossain (2013). *Using CORS*. Recuperat el 2 de Juny de 2014, desde http://www.html5rocks.com/en/tutorials/cors/

- C. Zakas, Nicholas (2010). *Cross-domain Ajax with Cross-Origin Resource*. Recuperat el 2 de Juny de 2014, desde http://www.nczonline.net/blog/2010/05/25/cross-domain-ajax-with-cross-origin-resource-sharing/

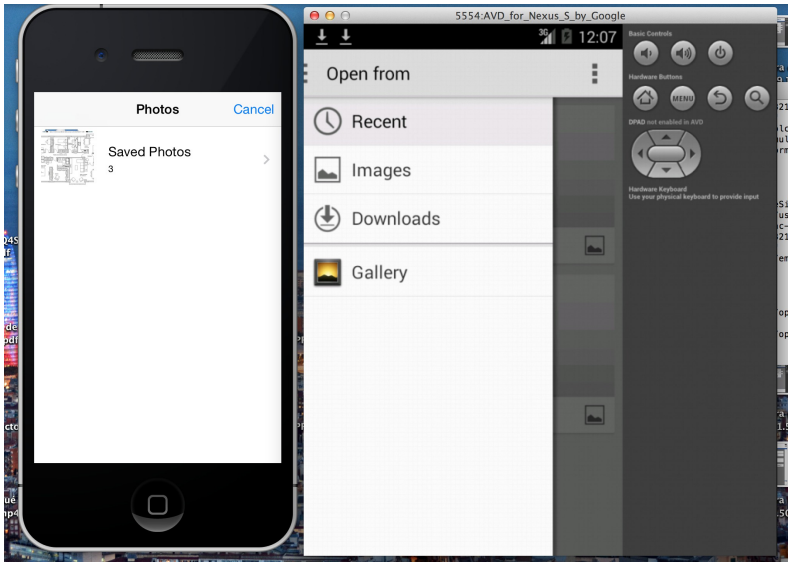
- Pérez Rivas, José Jesús (2013). *Como hacer un menú estilo Facebook.* Recuperat el 20 de Maig de 2014, desde http://www.phonegapspain.com/tutorial/como-hacer-un-menu-estilo-facebook/

7 Guia ràpida d'ús

Inici

Per començar cal obrir l'aplicació i introduir les credencials i la URL o IP del dispositiu ODControl:

A continuació en prémer el botó *LOAD&START* ens demanarà el plànol de l'habitacle a controlar:



Un cop seleccionada la imatge del mapa del carret del nostre dispositiu, ens mostrarà el mapa carregat per pantalla i podrem comprovar que ens podem moure sobre ell, fer zoom-in i zoom-out.

Ubicar ports

En aquest moment ja podrem ubicar ports sobre el mapa. Per fer-ho haurem de prémer el botó de la part superior esquerra i apareixerà de manera lliscant, el menú principal a través del qual podrem seleccionar els ports existents visibles al nostre dispositiu:

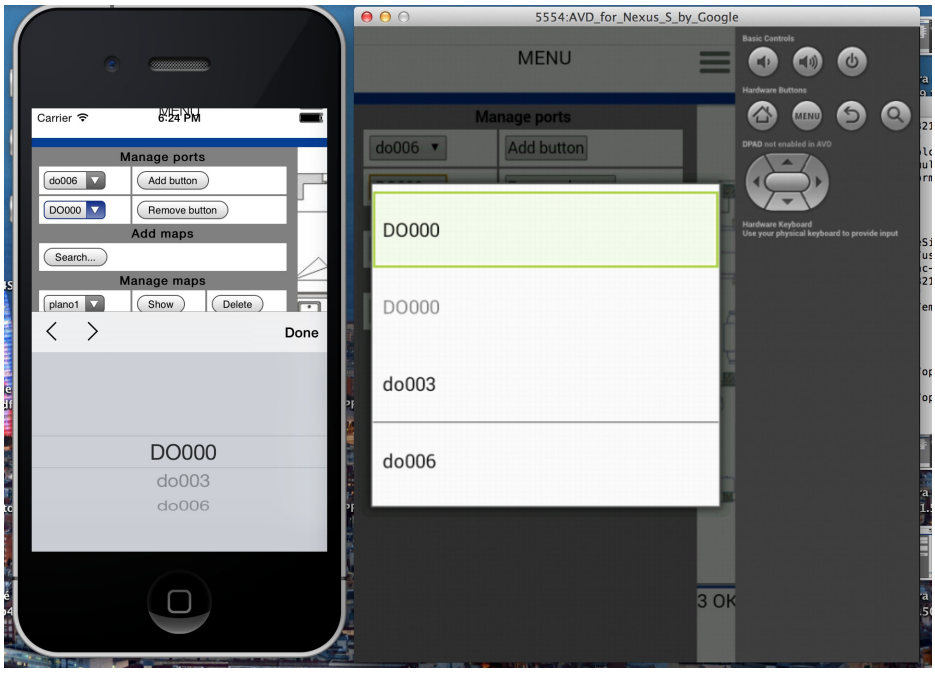
Per anar ubicant els ports només caldrà seleccionar el port desitjat i prémer *Add button.* Automàticament l'aplicació ens portarà al mapa i ens demanarà que mantinguem polsat sobre el mapa el dit durant 2 segons per col·locar el port.

Enviar ordres

Per commutar els ports digitals de sortida només cal mantenir polsat sobre l'interruptor 2 segons. Per als ports analògics de sortida seleccionem la imatge del port i assignem un valor.

Eliminar ports

Una vegada ubicats els ports, el desplegable adjunt al botó *Remove button,* s'omple amb els ports que s'han anat ubicant sobre el mapa. Si volem eliminar un, només cal que el seleccionem i polsem el botó. L'aplicació ens portarà automàticament al mapa i ens retornarà un missatge a través del terminal.

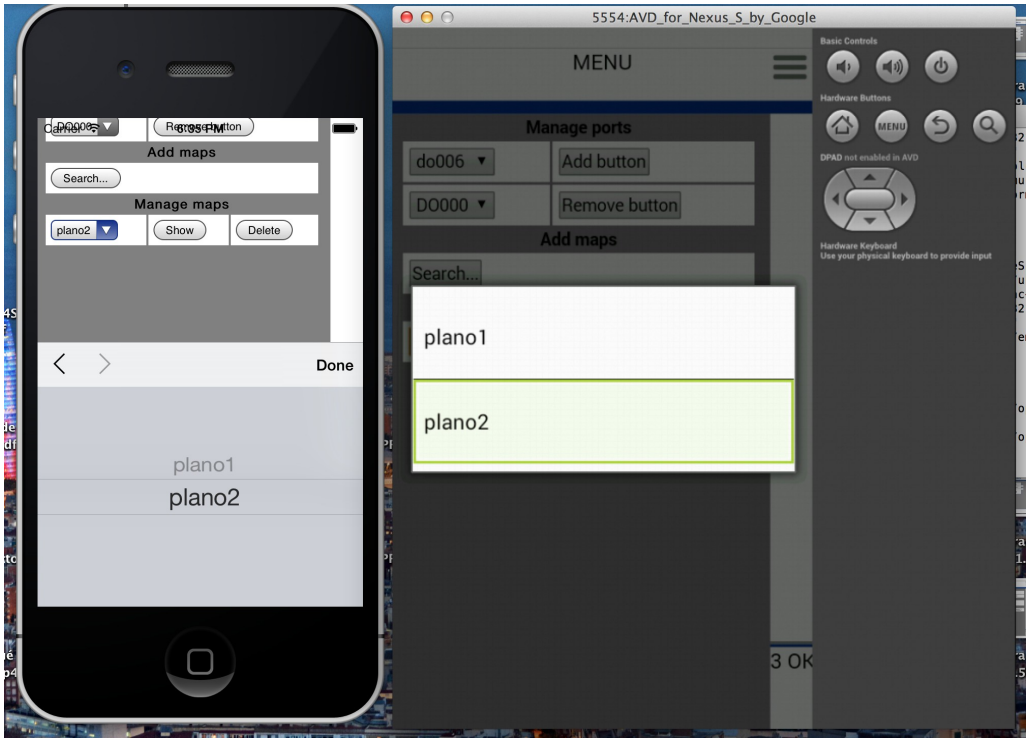


Administrar plantes

Si el que volem es afegir una nova planta només cal prémer el botó a la subsecció *Add Maps,* anomenat *Search...* . El funcionament serà exactament igual que al carregar el primer mapa.

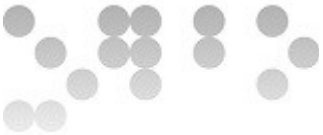
Cada vegada que afegim una planta, aquesta és carregada a la pantalla principal i al desplegable de la secció *Manage maps.* De manera que si volem moure'ns entre ells, caldrà seleccionar el mapa desitjat i apretar el botó *Show.*

En cas de voler esborrar el mapa, haurem de prémer *Delete*.



Preferències

A la part superior dreta de la pantalla principal, tenim un botó que ens porta a un menú que ens mostra algunes funcionalitats de configuració:



8 GNU Free Documentation License

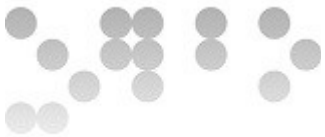
Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this



License principally for works whose purpose is instruction or reference.

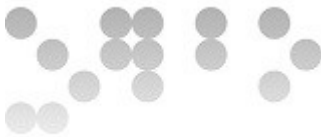
1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant



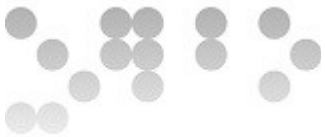
Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following



pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

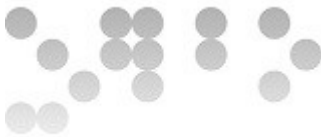
The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must



also follow the conditions in section 3.

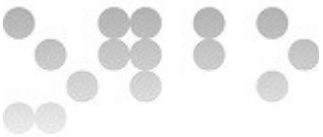
You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least



one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

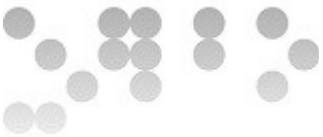
A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.



F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

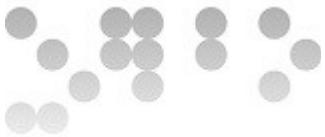
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.



If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

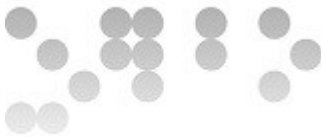
You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined



work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

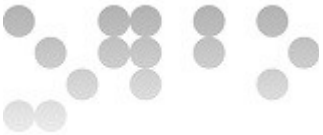
In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS



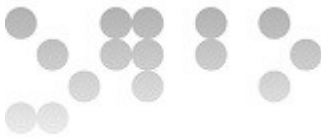
A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically



require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

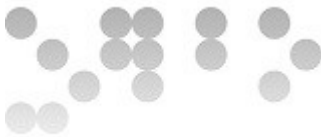
However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or



concerns. See http://www.gnu.org/copyleft/.

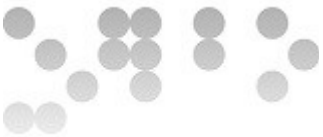
Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.



An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.