Sistema autònom d'alerta i control per a sala de servidors

Carles Caparrós Hernàndez
Grau de Tecnologies de la Telecomunicació
Arduino

Consultor: Antoni Morell Pérez
Professor: Pere Tuset Peiró

06/2017

Llicències alternatives (triar alguna de les següents i substituir la de la pàgina anterior)

A) Creative Commons:

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-CompartirIgual 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-SenseObraDerivada 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-CompartirIgual 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement 3.0 Espanya de Creative Commons

B) GNU Free Documentation License (GNU FDL)

no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU Free Documentation License".

C) Copyright

FITXA DEL TREBALL FINAL

Títol del treball:	*Sistema autònom d'alerta i control per a sala de servidors*
Nom de l'autor:	*Carles Caparrós Hernàndez*
Nom del consultor/a:	*Antoni Morell Pérez*
Nom del PRA:	*Pere Tuset Peiró*
Data de lliurament (mm/aaaa):	*06/2017*
Titulació o programa:	*Grau de Tecnologies de la Telecomunicació*
Àrea del Treball Final:	*Arduino*
Idioma del treball:	*Català*
Paraules clau	*Arduino*

Resum del Treball

Aquest projecte neix de la voluntat de poder disposar d'alertes que informin als interessats quan a una sala de servidors d'una oficina s'ha produit un tall de subministrament elèctric o hi ha hagut un pujada excessiva de la temperatura ambient. Aquests servidors estan permanentment encesos, donant-se el fet que hi ha períodes de temps que no tenen cap vigilància (nits, caps de setmana, vacances, ...).

Com a possible solució a aquesta necessitat s'ha implementat, mitjançant la integració de diferents dispositius del sistema Arduino, un sistema d'alertes per SMS.

Com a objectiu secundari, s'han aprofitat les capacitats dels components d'aquest sistema per tal de poder recollir i disposar d'un històric online de l'evolució de la variació de la temperatura a la sala i de les hores de funcionament de l'aire condicionat. L'anàlisi d'aquestes dades ha de permetre

saber si cal canviar l'ús que en l'actualitat se'n fa de la sala per tal de poder optimitzar-ne el consum energètic.

Abstract:

This project has born from the desire to provide alerts to inform those interested when a power cut has happened or there has been an excessive rise of the temperature in a room office servers. These servers are permanently lit, giving the fact that there are periods that have no monitoring (nights, weekends, holidays ...).

As a possible solution to this need, has been implemented through the integration of different system devices Arduino, an alert system SMS.

As a secondary objective, have been exploited the capabilities of the components of this system in order to collect and dispose of an online history of the evolution of the temperature variation in the room and hours of air conditioning running. The analysis of these data should allow knowing whether we need to change the use currently made of it in the room, in order to optimize energy consumption.

Índex

Índex d'il·lustracions

1. Introducció

1.1 Context i justificació del treball

Existeix avui dia una tendència a emmagatzemar la informació al núvol per tal de poder accedir a ella des de qualsevol lloc, a qualsevol hora i des de múltiples dispositius. Fins i tot, moltes aplicacions s'estan portant cap al núvol de manera que ni tant sols és necessari fer instal·lacions a les màquines locals.

Malauradament, però, aquesta forma de treballar requereix disposar d'una bona connectivitat a Internet - que no sempre està garantida a tot el territori- per tal d'accedir a la informació. Aquest inconvenient fa que a les xarxes locals en les que es necessita compartir arxius i documents –com ara a oficines o escoles-, seguint una política de permisos, encara sigui recomanable disposar de servidors locals que s'encarreguin de fer la validació d'usuaris i l'emmagatzematge de la informació. En aquelles feines en les que els treballadors necessiten o volen accedir remotament –des de fora de les a instal·lacions- a la informació que resideix dins d'aquesta xarxa, es fa necessari que aquests dispositius hagin d'estar permanentment encesos i que, per tant, no es puguin desconnectar en tancar l'oficina cada nit, ni els caps de setmana, ni en períodes vacacionals.

Cal doncs garantir el correcte funcionament dels aparells les vint-i-quatre hores del dia i els tres-cents seixanta-cinc dies de l'any. Això comporta que, a banda dels propis dispositius que formen el sistema informàtic, calgui disposar d'altres elements de protecció davant de possibles incidències com poden els sistemes d'alimentació ininterrompuda o SAI –per garantir el funcionament davant fallades de la xarxa elèctrica- o els aires condicionats -per mantenir la temperatura ambient de l'habitació on estan ubicats aquests dispositius.

Amb aquest projecte es pretén donar una solució per a aquells qui no volen arriscar-se a que una fallada en el sistema de climatització dels servidors pugui comprometre la informació resident en els mateixos. Si no es vol –o no es pot- disposar d'un segon aire condicionat que refrigeri la sala o habitació on els servidors treballen les 24 hores de tots el dies de l'any, amb el dispositiu dissenyat en aquest projecte, les persones seleccionades, podran rebre un avís informant de l'increment no desitjat de temperatura o del tall de subministrament elèctric per tal de poder solucionar el problema.

1.2 Objectius del treball

Al marge del cas general anteriorment descrit, concretant el cas de l'oficina per a la qual es dissenya el prototip, la sala on es troba l'armari amb els servidors i dispositius d'emmagatzematge de la xarxa, s'utilitza també com a arxiu de paper i també és el lloc on hi ha una fotocopiadora. Aquest fet comporta que la temperatura de l'habitació estigui sotmesa a constants canvis de temperatura i per tant l'aire condicionat no pari de treballar.

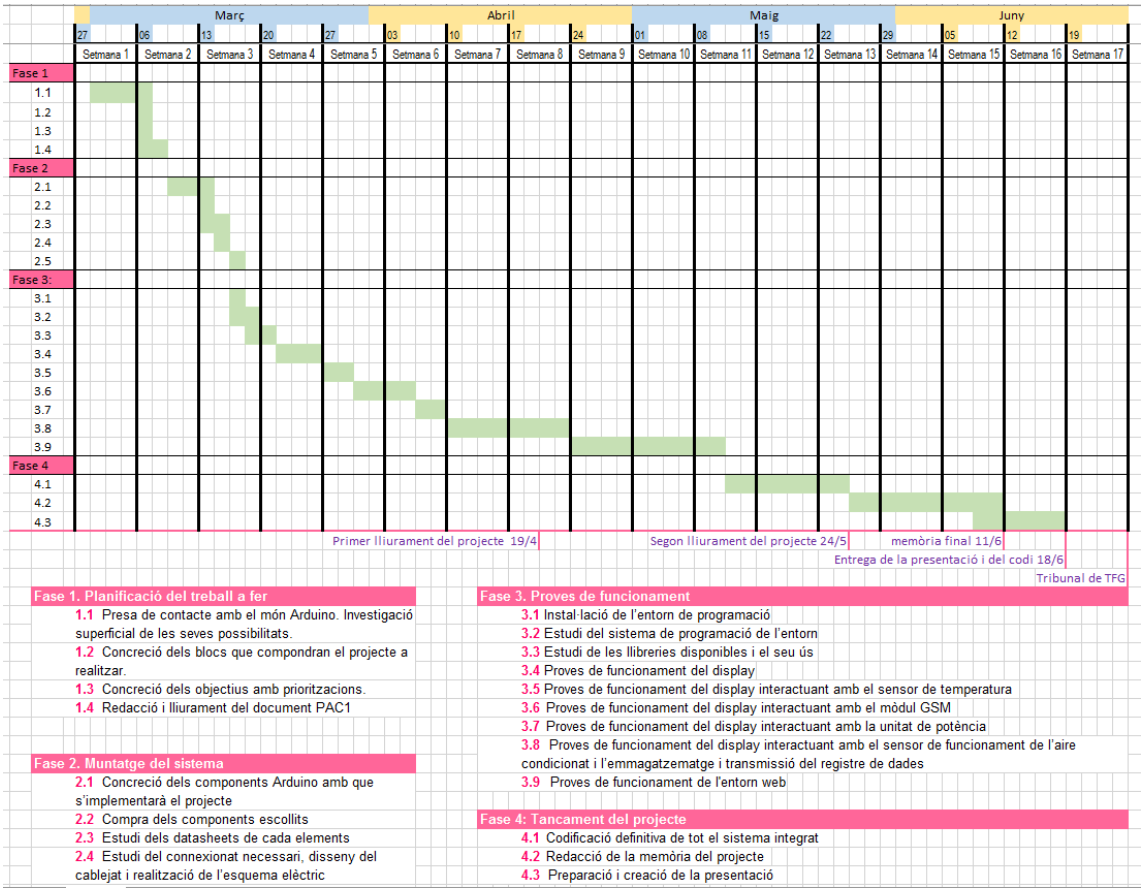
Amb tot això, els objectius que es persegueixen amb aquest treball són:

- Com a objectiu principal es pretén disposar d'un aparell que vetlli pel correcte funcionament del sistema de climatització de la sala i que enviï una alerta per SMS així que es detecti que la temperatura de la sala superi el llindar fixat, així com en el cas del tall del subministrament de la xarxa elèctrica.
- Com a segon objectiu, es desitja poder accedir a la informació de l'estat de les condicions de la sala remotament a través d'un entorn web.
- Com a tercer objectiu es vol disposar d'un registre de l'evolució de l'estat del sistema per tal que es pugui fer un anàlisi de la conveniència o no de modificar els usos o les instal·lacions de la sala.

1.3 Enfocament i mètode seguit

Per tal de realitzar aquest treball, tenint en compte els coneixements previs de l'autor, s'ha organitzat les tasques a fer en diferents fases:

- Fase 1. Planificació del treball a fer.
- Fase 2. Muntatge del sistema
- Fase 3. Proves de funcionament.
- Fase 4. Tancament del projecte.



Il·lustració 1 Planificació del treball

Cal destacar dos aspectes que han provocat una important modificació en aquesta planificació inicialment plantejada. El primer i principal ha estat que l'autor, per causes alienes a la seva persona, no ha pogut dedicar-hi temps al projecte durant un període destacable del temps

disponible. Això ha provocat un retard important que en gran part s'ha pogut compensar amb una dedicació molt més elevada durant les setmanes posteriors al conflicte d'interessos.

L'altre aspecte destacat que ha consumit moltes hores no planificades ha estat el pas del prototip funcional al prototip comercial. Aconseguir encabir el sistema en una caixa d'un volum adequat i amb unes connexions que permetin de forma fàcil l'assemblatge i des-assemblatge del components que integren el sistema ha estat una de les tasques més laborioses. Deixant els incidents a banda, cal dir que la resta de tasques s'han ajustat força al temps inicialment previst.

1.4 Sumari productes obtinguts

En finalitzar el treball s'haurà creat:

- un prototip comercial que permetrà enviar alertes SMS als números de telèfons desitjats si es produeix un tall del subministrament elèctric a la sala on s'ubica l'aparell o si, en aquesta mateixa sala, es produeix un increment de temperatura que supera el llindar desitjat.
- Un conjunt de planes web que permeten consultar l'històric de l'evolució de la temperatura de la sala i de l'aire condicionat registrada per sistema dissenyat.
- Un programa creat en codi compilable per Arduino que gestiona els diferents elements del sistema.
- Aquest document i una presentació en vídeo que el complementa.

2. Ha calgut fer

En aquest apartat es destaca allò que ha calgut fer obtenir els productes abans esmentats en el punt 1.4.

A l'inici del projecte, l'autor del treball, tot i que n'havia sentit parlar força, no havia tingut cap contacte amb el món Arduino. Així doncs ha calgut fer una recerca intensiva d'informació per la xarxa per esbrinar el seu funcionament. Ha calgut també passar del món teòric a la realitat i practicar amb un parell d' *starter kits* per tal d'anar agafant confiança amb aquest món. Cal dir que seguint les receptes es pot aprendre molt sobre el funcionament d'aquest sistema

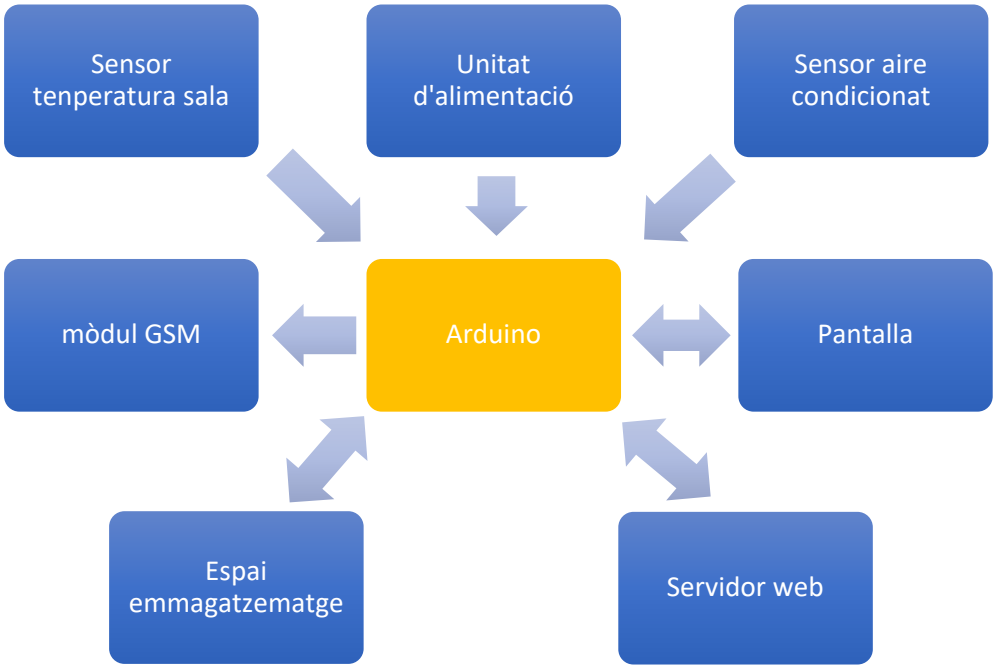
Amb els nous coneixements adquirits i els objectius del que es vol disposar en finalitzar el projecte, sembla fàcil poder triar els components a comprar per tal d'integrar-los en un sistema. A l'hora de la veritat aquesta tasca és més complicada del que sembla ja que cal llegir molta informació per tal de veure què es compatible, què és necessari i que s'ha de descartar per tal que els costos del producte no es disparin en excés.

Així doncs el que veurem als propers apartats serà:

- El DIAGRAMA DE BLOCS DEL SISTEMA pel que fa al maquinari que formarà part del sistema.
- Els ELEMENTS DEL MAQUINARI que formaran cadascun dels blocs del diagrama.
- Les INTERCONNEXIONS, cablejades o no, que es necessiten per tal que els diferents elements del sistema puguin interactuar entre ells.
- El PROGRAMARI i el codi que s'ha escrit per tal que el sistema, un cop muntat, faci la feina que s'espera d'ell.

2.1 Diagrama de blocs del maquinari

Conceptualment, el mapa de blocs del sistema es pot representar de la següent manera:



Il·lustració 2 Diagrama de blocs del maquinari

2.2 Els elements del maquinari

En aquest apartat es descriuen les principals característiques dels elements triats per dur a terme funcions de cada bloc. En els apartats 2.3 i 2.4 s'explicaran com s'han fet les connexions i com s'han programat els mateixos.

2.2.1 L' Arduino Yún

Aquesta és una placa avançada de Arduino que té com a característica principal que incorpora connectivitat Ethernet i Wi-Fi i una ranura per targeta micro-SD (a la part de sota) . A més també incorpora un distribució Linux. S'ha triat aquesta per la gran capacitat de connectivitat així com per disposar de capacitat d'emmagatzematge per desar les

dades que va recollint el sistema. Així mateix, la distribució Linux obre un món de possibilitats a l'hora d'augmentar les funcionalitats del sistema.

Així doncs, amb aquest element cobrim tres dels blocs identificats al mapa de blocs de la il·lustració 2: l'Arduino, l'espai d'emmagatzematge i el servidor de planes web.

Il·lustració 3 Vista superior placa Arduino

Pel que fa als aspectes elèctrics, aquesta placa només es pot alimentar a través del port micro-USB o a través del pin V_{IN} . Dit de una altra manera, aquesta placa no té una entrada específica per a la alimentació. A més, també a diferència d'altres plaques Arduino, necessita que aquesta alimentació sigui de no superior al 5V i que estigui estabilitzada. Altres plaques de la família es poden alimentar a més de 5V ja que disposen d'un convertidor intern que baixa la tensió d'entrada a 5V des de tensions inferiors als 12V -tot i que no es recomana alimentar-les a aquesta tensió durant un interval massa llarg de temps-.

Segon les especificacions del fabricant, aquesta placa consumeix 250mA quan està en funcionament.

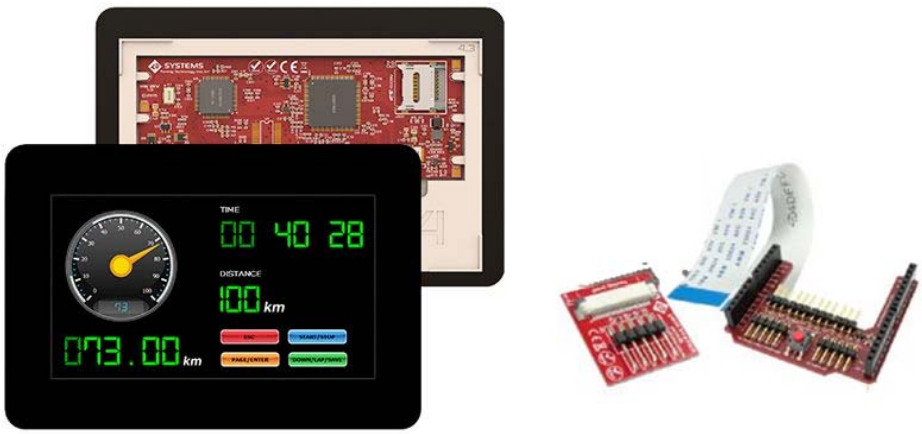
2.2.2 LA PANTALLA

En aquest apartat l'autor s'ha permès optar per a una pantalla de prestacions molt més elevades a les que requereix el treball. El motiu d'aquesta elecció és que es pretén utilitzar la mateixa per a futures ampliacions de les prestacions del dispositiu dissenyat.

En comparació amb els displays i les pantalles que s'utilitzen habitualment en els projectes amb Arduino, s'ha buscat una pantalla de grans dimensions i que fos tàctil per tal de poder interactuar amb els dispositiu sense necessitat de connectar-se a través d'un ordinador.

La pantalla seleccionada pertany a la companyia 4D Systems i té com a referència gen4-uLCD-43DCT-CLB.

Té una resolució de 480x272 píxels i unes dimensions de 4,3". Per tal de simplificar la interconnexió amb les plaques d'Arduino, es disposa d'un escut per Arduino dissenyat per proporcionar una interfície de connexió senzilla entre les pantalles 4D i les plaques. Tal i com veurem més endavant, no és obligatori utilitzar aquest escut ja que es poden connectar directament els pins adients de la placa amb l'adaptador que proporciona la pantalla.



Il·lustració 4 Pantalla LCD de 4D Systems

Més endavant veurem que aquesta pantalla no es programa de l'entorn d'Arduino sinó que té el seu propi entorn de configuració i programació. Aporta uns gràfics d'un alt nivell de qualitat.

El consum típic d'aquesta pantalla és de 350mA amb el contrast fixat al seu màxim.

2.2.3 El mòdul GSM

Per fer l'enviament dels SMS s'ha triat el mòdul SIMCOM GSM GPRS de Kuman basat en l'integrat SIM900. Tot i que en aquest projecte només s'utilitzarà el mòdul per a l'enviament dels missatges de text, també permet la realització i recepció de trucades telefòniques i la recepció de missatges SMS.

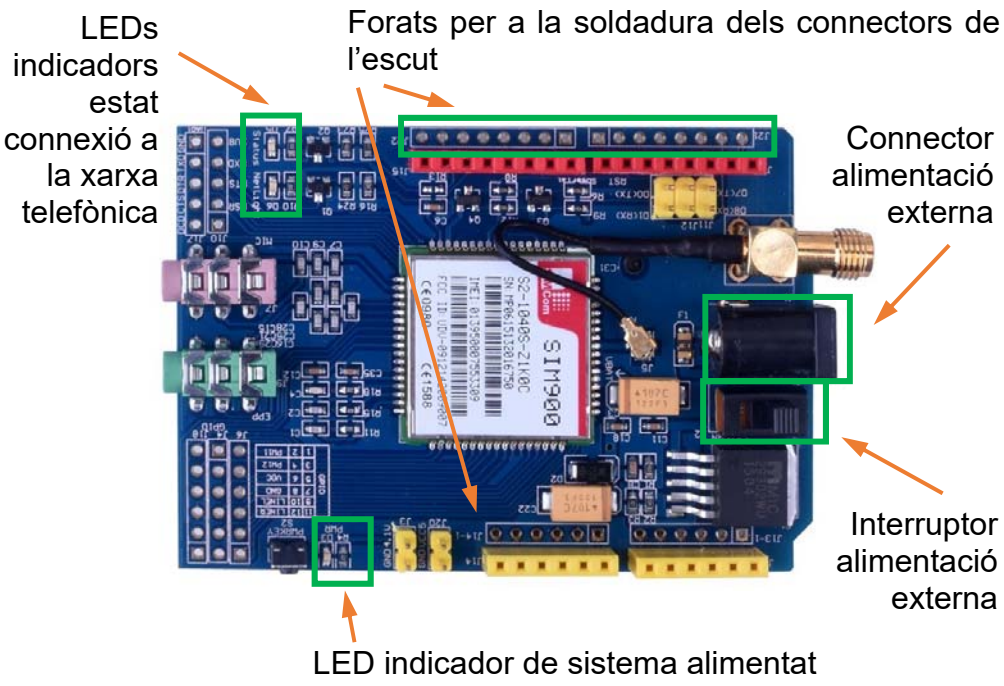
Il·lustració 5 Mòdul GSM

Tot i que inicialment el dispositiu no ve preparat per a poder-ho connectar com a un escut de l'Arduino, el mòdul ve preparat amb els forats en el lloc adequat per tal de poder soldar els connectors –que també arriben amb el *pack*- que permeten utilitzar-ho com un escut més. I això és el que s'ha fet.

Segons les especificacions, el mòdul pot arribar a consumir un intensitat d'uns 2A per la qual cosa li cal una alimentació externa. Per sort la placa ja ve preparada amb un connector estàndard d'alimentació i un

commutador que permet seleccionar si l'alimentació ve a través dels pins de l'escut o de l'alimentació externa.

En les proves que s'han fet s'ha comprovat que malgrat el mòdul arrenca normalment amb l'alimentació provinent dels pins de l'escut, quan la placa intenta fer el registre en la xarxa telefònica es produeix un *reset* de la mateixa quan no es disposa d'alimentació externa.



Il·lustració 6 Vista superior mòdul GSM

Si el mòdul està ja connectat a la xarxa telefònica o bé està buscant senyal per tal de poder-se registrar en ella ho sabem de manera visual per la freqüència amb la que s'encenen o s'apaguen els lums LED que indiquen l'estat d'aquesta connexió (veure il·lustració 5). Una encesa i apagada cada:

- 0,8 segons ⇒ indica que està cercant la xarxa per a registrar-se.
- 3 segons ⇒ indica que està correctament registrat a la xarxa i que per tant està en disposició d'enviar missatges de text quan sigui necessari.

2.2.4 LA FONT D'ALIMENTACIÓ

Els requeriments dels 5V estabilitzats de la placa Arduino Yún ha fet necessari disposar d'una font que compleixi aquesta limitació. Sabent el consum de 2A de pic que el mòdul GSM pot arribar a assolir, la font d'alimentació triada ha estat una de l'empresa Mean Well amb referència RS-25-5. Aquesta font de preu molt reduït disposa de protecció per curtcircuits, sobrecàrregues i sobretensions. Ofereix una sortida màxima de 5A d'intensitat que és suficient por suportar el consum màxim de tot el sistema.

Il·lustració 7 Font d'alimentació

2.2.5 LES BATERIES

En el moment de l'elecció de la bateria, el primer que s'ha mirat ha estat la tecnologia amb que s'ha fabricat. Donat que el funcionament d'aquest sistema amb l'energia provinent de la bateria només està prevista per a casos d'emergència, s'ha cregut oportú triar bateries del tipus LiPo que tenen com a característica destacable el fet que no perden tensió amb al pas del temps si no es fa ús d'elles. Cercant una bateria que permeti mantenir el sistema en funcionament durant un tall típic de subministrament -que normalment no passa de la mitja hora- s'ha triat una bateria de 7,4V i de 2000mAh. En el moment de la compra a més s'ha trobat un *pack* que a un preu raonable ofereix dues bateries d'aquestes característiques juntament amb el seu carregador.

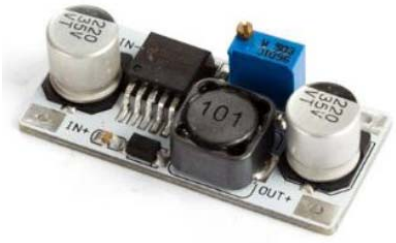
Il·lustració 8 Bateries LiPo i carregador

El carregador permet la càrrega simultània de dos d'elles i a més disposa de protecció contra sobrecàrregues i curtcircuits.

Per incrementar el temps amb que el sistema pot funcionar amb les bateries el que s'ha fet és posar les dues bateries en paral·lel.

2.2.6 MÒDUL CONVERTIDOR DC-DC *STEP DOWN*

Donat que la tensió nominal de les bateries és de 7,4V i que ni l'Arduino ni la pantalla tàctil accepten aquestes tensions, ha calgut rebaixar aquesta tensió fins els 5V mitjançant un mòdul convertidor DC-DC step down.

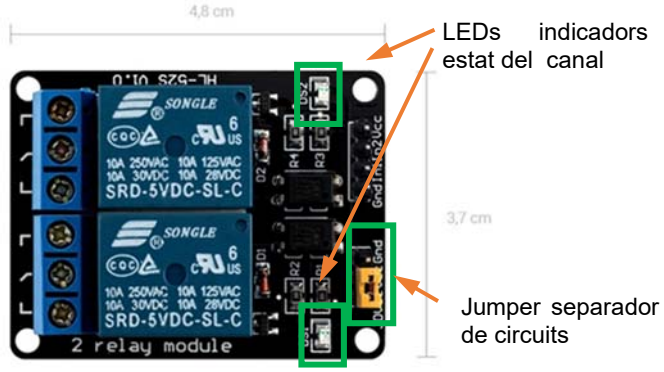


Il·lustració 9 Mòdul convertidor step down

El mòdul seleccionat a estat el VMA404 de la casa Velleman, basat en el regulador LM2596S.

2.2.7 MÒDUL RELÉ

Com a mecanisme commutador entre l'alimentació a partir de la xarxa elèctrica i l'alimentació a partir de les bateries s'ha decidir utilitzar un relé. Malgrat només es necessita un per a fer aquesta funció s'ha utilitzat un mòdul que n'incorpora dos.



Il·lustració 10 Mòdul relé de 2 canals

Tal i com es destaca a la il·lustració anterior, aquest mòdul incorpora un led extern pes a cascun dels canals que informa visualment de si el relé està activat o en mode repós.

També, mitjançant un jumper -destacat a la **¡Error! No se encuentra el origen de la referencia.** en color groc-, disposa de l'opció d'independitzar totalment els dos circuits que formen part de l'etapa de control: el que alimenta l'optoacoblador i el que alimenta la bobina. En aquest projecte no es fa ús d'aquesta possibilitat.

El mòdul seleccionat és ideal per utilitzar amb l'Arduino ja que el corrent consumit per al control de l'estat és d'un màxim de 20mA, que és la meitat de la intensitat que permet circular qualsevol dels pins de l'Arduino.

A més, aquest mòdul disposa d'un optoacoblador que permet aïllar el circuit de control de possibles problemes del circuit controlat.

2.2.8 SENSOR DE TEMPERATURA

Com a sensor de temperatura s'ha escollit el sensor LM35 ja que el seu rang de mesures va des del 0 fins els 100 graus centígrads –més que suficient per les temperatures que s'espera mesurar- i la seva sortida varia 10mV per cada grau centígrad, cosa que facilita molt la conversió de la mesura elèctrica que sap gestionar l'Arduino a la mesura tèrmica que volem controlar. La seva precisió és d'aproximadament mig grau .

Il·lustració 11 Sensor LM35

D'aquest element se n'utilitzaran 2, un per mesurar la temperatura de la sala i un altre per mesurar la temperatura de l'aire que està sortint en cada moment de l'aire condicionat. Ha passat més d'una vegada que l'aire condicionat ha estat encès però per un problema amb el refrigerant no ha estat capaç de refredar la sala. És per això que no hi ha prou de saber si l'aire condicionat està encès sinó que a més també cal prendre aquesta segona lectura de temperatura.

Per portar tant l'alimentació com el senyal de tornada s'ha decidit utilitzar un cable de xarxa donat que és quelcom abundant en el lloc on està destinat a anar i, a més, dona flexibilitat de poder situar l'aparell en una posició més propera o més allunyada de la sortida de l'aire condicionat.

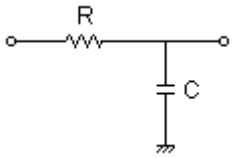
Un problema que ha aparegut en el moment d'abandonar la *protoboard* i passar a un prototip comercial ha estat que el mesurament de la temperatura que arribava a l'Arduino no era correcta per culpa de les interferències electromagnètiques. Resulta que el voltímetre si que

donava el valor correcte però les lectures que s'obtenien a l'Arduino eren totalment erràtiques.

Es va decidir aleshores que calia posar un filtre passabaixes per acabar amb el problema i es va dissenyar un filtre RC amb una freqüència de tall de 10Hz.

Per determinar el valors del la resistència i el condensador a utilitzar en el filtre es va consultar per internet i es va trobar la plana de Okawa Electric Design que facilitava aquesta tasca (OKAWA Electric Design, 2017). A partir de la freqüència de tall desitjada i el valor d'un condensador, la plana et proposa el valor de la resistència a utilitzar i ofereix molta informació sobre el comportament del filtre (Funció de transferència, freqüència de tall, diagrama de Bode ,...).

Així doncs, el sistema disposa de dos filtres RC (una per a cada sensor de temperatura) amb els següents valors:

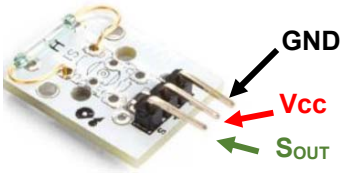


f_c=10,26Hz
C=33 µF
R=470 Ω

Il·lustració 12 Filtre passa baixes per sensors temperatura

2.2.9 SENSOR *REED* DE CONTACTE

Com a sensor de contacte per saber si l'aire condicionat està funcionant o no s'ha escollit un sensor reed. Aquest element és un interruptor normalment obert que es tanca en presència d'un camp magnètic i que ja incorpora una resistència de pull-up de 10KΩ que permet connectar-la directament a l'entrada de l'Arduino.



Il·lustració 13 Sensor Reed

Amb aquest sensor el que es determina és si la comporta de l'aire condicionat està oberta o tancada. Donat que en els *splitters* aquesta comporta es tanca quan l'aire condicionat no està en funcionament, aquest sensor ens informarà de manera indirecta de l'estat de l'aire condicionat.

Tal i com s'ha explicat en el punt anterior, la comunicació d'aquest sensor al el dispositiu central es durà a terme mitjançant un cable de xarxa.

2.3 La interconnexió elèctrica

Interconnectar tots els elements del circuit d'una de forma endreçada i estable no ha estat gens fàcil. Un dels avantatges del sistema Arduino i les *protoboards* és que és relativament senzill anar construint un sistema a base d'anar afegint cables de connexió entre un lloc i un altre i establint ponts a la *protoboard* allà on calgui.

Aquest avantatge es converteix en un inconvenient quan el sistema es va fent gran donat que també és fàcil que es produeixi una desconnexió involuntària de cables i després cal trobar quin és el seu lloc.

Altre aspecte important és la qualitat dels cables i connectors. Sobretot pel que fa als cables que porten l'alimentació als diferents dispositius. Durant la implementació del sistema han aparegut problemes amb mòduls que deixaven de funcionar correctament i que després de moltes proves ha estat un problema dels cables emprats. Per exemple, utilitzant els cables acabats amb connectors de tipus Dupont de que es disposava per alimentar el mòdul GSM, aquest no era capaç de registrar-se a la

xarxa. En canviar els cables, per altres de millor qualitat, els sistema funcionava correctament. Així doncs, per totes les connexions d'alimentació dels diferents mòduls s'han utilitzant uns cables diferents dels que s'han utilitzant per intercanviar dades o alimentar els sensors.

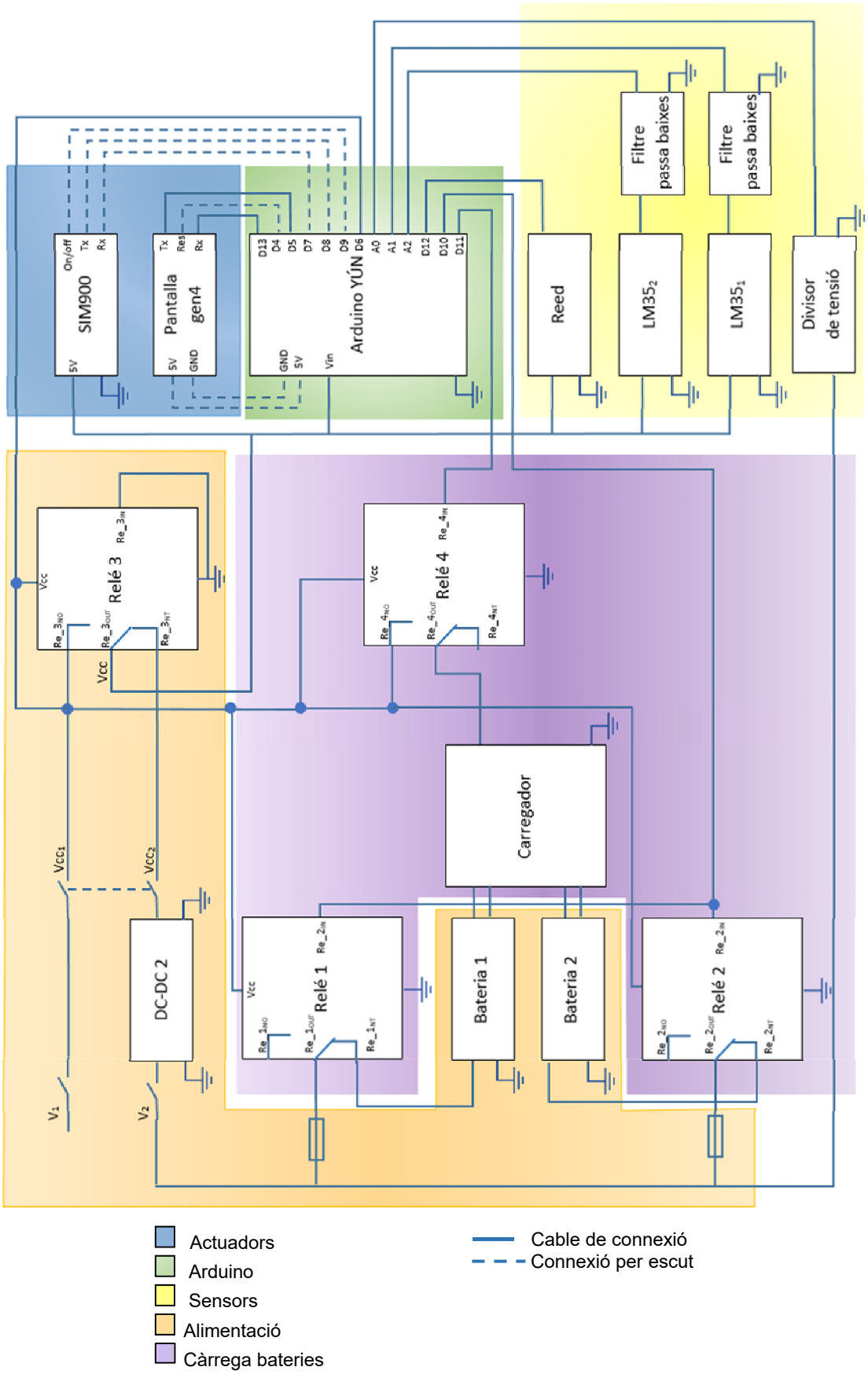
Il·lustració 14 Connectors tipus Dupont

Per a totes les connexions emprades s'ha utilitzat o bé connectors tipus Dupont o bé soldadures, a banda d'aquestes connexions que ja disposen un regletes de connexions (els relés).

S'ha procurat agrupar els connectors en conjunts degudaments etiquetats i formant combinacions que dificultin l'error si cal desmuntar el dispositiu i tornar-lo a muntar

A la il·lustració 15 es pot veure l'esquema elèctric de tot el muntatge i tot seguit s'explicarà en detall els aspectes més destacats.

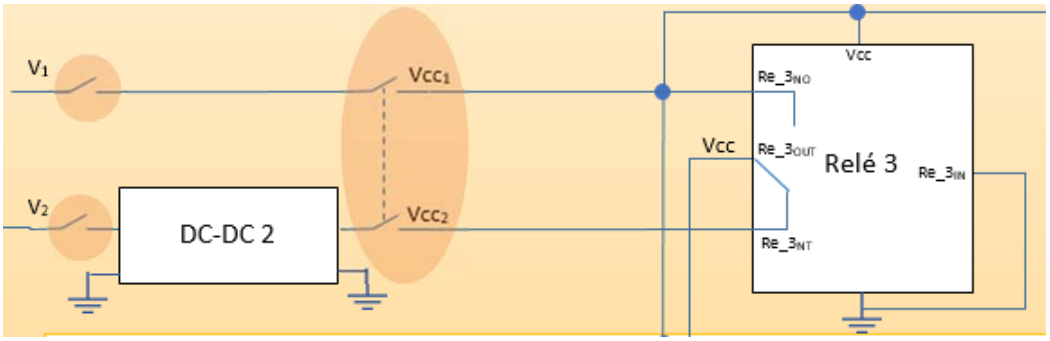
2.3.1 ESQUEMA ELÈCTRIC



Il·lustració 15 Esquema elèctric

2.3.2 LES CONNEXIONS DE FONTS D'ENERGIA

El sistema disposa de dues fonts d'energia: la provinent de xarxa elèctrica –que anomenarem Vcc1– i la provinent de les bateries –que anomenarem Vcc2-. La tensió provinent de la xarxa elèctrica no necessita cap adaptació ja que prové d'una font d'alimentació de 5V estabilitzada. La tensió provinent de les bateries és el resultat d'haver adaptat els 7,4V nominals d'aquestes als 5V requerits pel sistema gràcies a un mòdul convertidor DC-DC *step down*.



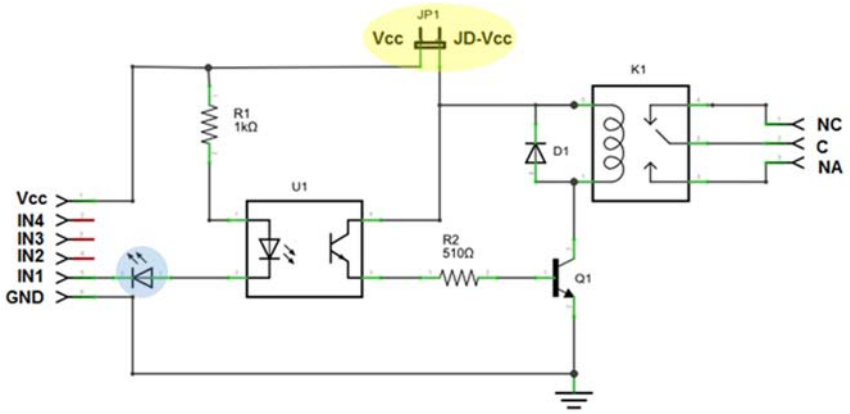
Il·lustració 16 Selector font d'alimentació

El dispositiu disposa d'un interruptor -accessible des de l'exterior- que talla alhora les dues font d'energia.

Il·lustració 17 Detall interruptor exterior

Per tal de poder aïllar les dues fonts durant la fase de proves, s'han afegit al circuit un jumper per a cadascuna de les fonts que permet desconnectar el circuit en qüestió.

El relé 3 és l'encarregat de decidir quina de les dues fonts és la que utilitza el sistema. Com es pot veure a la il·lustració 18, s'aprofita la tensió Vcc1 per alimentar directament per alimentar-lo. El funcionament del relé es mostra a la següent figura:

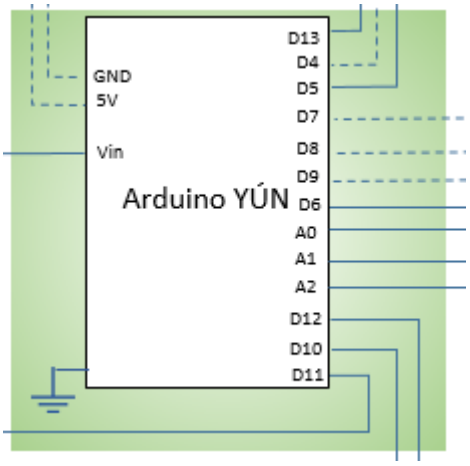


Il·lustració 18 Funcionament del relé

Es pot comprovar com quan a Re_3IN tenim 0V, aleshores hi ha caiguda de tensió a l'optoacoblador i per tant aquest excita el transistor intern del mateix que al seu torn excita el transistor extern que fa circular corrent per la bobina que al seu torn activa realment el relé.

És per això que el relé 3 té connectada aquesta entrada de control directament a terra. D'aquesta manera, sempre que hi hagi alimentació de la xarxa elèctrica el relé 3 estarà activat i per tant en consumirà l'energia provinent d'aquesta. En el moment que falli l'alimentació de la xarxa elèctrica, el relé 3 passarà a mode repòs i en aquell moment es passarà a consumir l'energia provinent del circuit de les bateries.

2.3.3 LES CONNEXIONS DEL MICROCONTROLADOR



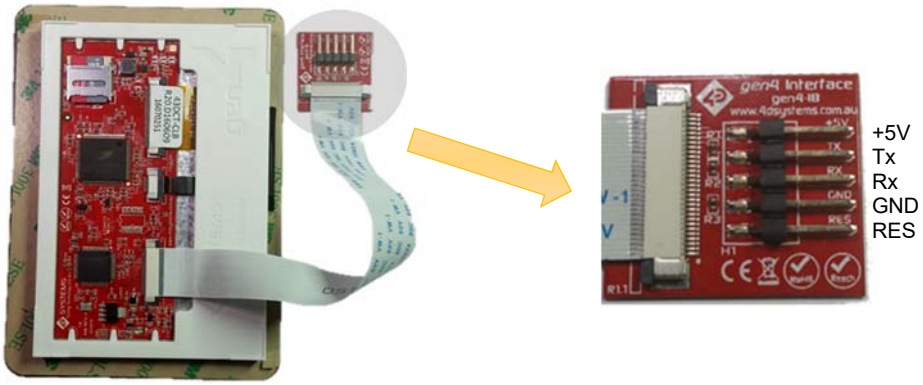
PIN	FUNCIÓ	DISPOSITIU
D13	Rx	pantalla
D4	Reset	
D5	Tx	
D7	Rx	SIM900
D8	Tx	
D9	On/Off	
D6	Vcc_1	Sensors
A0	V_{BAT}	
A1	$LM35_1$	
A2	$LM35_2$	
D12	Reed	
D10	Relè 4	
D11	Reles 1 i 2	

Il·lustració 19 Connexions del microcontrolador

Tal i com s'aprecia a la il·lustració anterior, a banda dels pins V_{IN} i GND usats per alimentar l'Arduino, s'han usat 3 pins analògics i 10 de digitals. Les línies discontínues indiquen que no hi ha un cable físicament connectat al pin sinó que la connexió es fa a través dels propis connectors que interconnecten els *shields*.

2.3.4 Les connexions de la pantalla tàctil

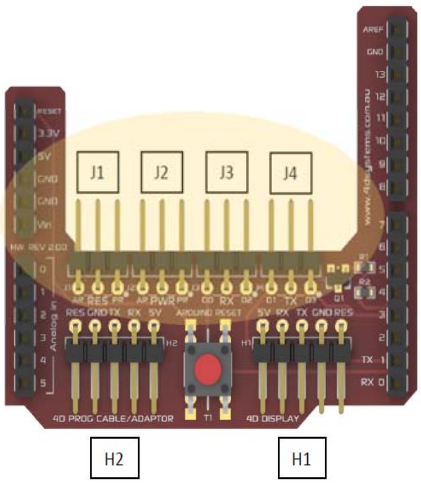
El primer escut punxat sobre l'Arduino és el *shield* de la pantalla. Tal i com es pot veure a la il·lustració 20, tant l'alimentació com la informació li arriben a través d'un cable pla de 30 línies. A l'altre extrem del cable trobem una placa adaptadora que finalment ens dóna accés a 5 pins de connexions. A la imatge ampliada del connector es pot observar com, a banda dels dos pins per l'alimentació, n'hi ha dos més destinats a la comunicació sèrie de dades i un darrer pin que serveix per a fer un *reset* de la pantalla



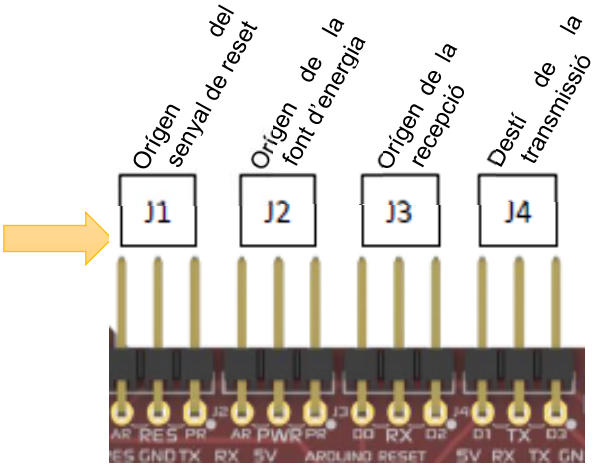
Il·lustració 20 Connexions de la pantalla tàctil

Establint la connexió des d'aquests terminals directament fins a l'Arduino permetria realitzar estalviar-se d'utilitzar el corresponent escut adaptador. L'escut, però, facilita la interconnexió entre la pantalla i el mircrocontrolador ja que els cables arriben d'una forma més endreçada.

A la il·lustració 21 veiem com és aquest escut. Els 5 cables provinents del connector de la pantalla es connecten al conjunt de pins etiquetats com a H1 (*Header* 1). El conjunt de pins etiquetats com H2 (*Header* 2) bàsicament permeten alimentar la pantalla des d'una font d'energia externa a l'Arduino. Els conjunts de pins etiquetats com a J1, J2, J3 i J4 permeten modificar el comportament de l'escut gràcies a la utilització de *jumpers*. El botó vermell serveix per a fer un reset de la placa Arduino, no de la pantalla. Això és degut a que en algunes plaques l'accés al botó de reset queda dificultat per la presència de l'escut.



Il·lustració 21 Connexions de l'escut de la pantalla



Il·lustració 22 Detall del jumpers de l'escut de la pantalla

El *jumper* J1 permet triar d'on arriba el senyal de reset cap a la pantalla. Segons la posició del *jumper* tenim:

- ⇒ L'ordre de *reset* prové de la placa Arduino a través del pin D4.
- ⇒ L'ordre de *reset* prové del pin corresponent del connector H2
- ⇒ No es pot fer reset de la pantalla

Per a aquest projecte s'ha triat la primera posició, és a dir, l'ordre de *reset* l'haurà de donar l'Arduino a través del pin D4.

El *jumper* J2 permet triar l'origen de la font que dóna energia a la pantalla. Segons la posició del *jumper* tenim:

 ⇒ L'energia prové de la placa Arduino a través del pin de 5V

 ⇒ L'energia prové del pin corresponent del connector H2

 ⇒ No arriba energia a la pantalla i no es podrà encendre

Per a aquest projecte s'ha triat la primera posició i per tant l'energia a la pantalla li arriba a través del propi arduino. D'aquesta manera evitem dos cables entre l'escut i una altra font d'energia. S'ha comprovat que l'arduino pot alimentar aquesta pantalla sense problemes.

El *jumper* J3 permet triar l'origen de les dades que arriben a la pantalla a través de la comunicació sèrie. Segons la posició del *jumper* tenim:

 ⇒ La transmissió arriba des del pin D0 de la placa Arduino

 ⇒ La transmissió arriba des del pin D2 de la placa Arduino

 ⇒ La transmissió arriba des d'allà on es connecti el pin central del *jumper*.

En el cas de l'Arduino Yún el pin D0 (serial Rx) està ocupat per la comunicació entre el microcontrolador (Atmega32U4) i el microprocessador (Atheros AR9331) que formen part d'ell. En aquest projecte s'ha optat per utilitzar la darrera opció i el pin central es connecta a al pin D13.

El *jumper* J4 permet triar el pin de destí de les dades provinents de la pantalla a través de la comunicació sèrie. Segons la posició del *jumper* tenim:

 ⇒ La transmissió es condueix cap el pin D1 de la placa Arduino

 ⇒ La transmissió es condueix cap el pin D2 de la placa Arduino

 ⇒ La transmissió es condueix cap allà on es connecti el pin central del jumper.

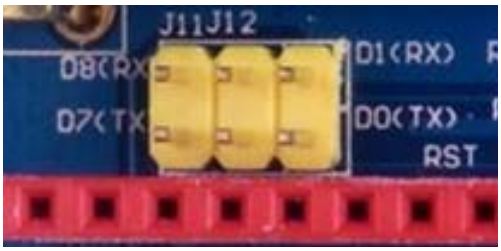
En el cas de l'Arduino Yún el pin D1 (serial Tx) està ocupat per la comunicació entre el microcontrolador (Atmega32U4) i el microprocessador (Atheros AR9331) que formen part d'ell. En aquest

projecte s'ha optat per utilitzar la darrera opció i el pin central es connecta a al pin D5.

2.3.5 Les connexions del mòdul GSM

A sobre de l'escut de la pantalla es troba situat l'escut que permet la comunicació via la xarxa de telefonia. Aquest escut pot arribar a tenir un consum de fins a 2A treballant a 5V per la qual cosa necessita una font d'energia externa a l'Arduino. És per això que a l'esquema es mostra com aquest mòdul va connectat directament a la font d'energia operativa en cada moment (la provinent de la xarxa elèctrica o la provinent de les bateries). S'ha comprovat que si s'alimenta directament a través del pin de 5V de l'Arduino, s'encén i comença a cercar la xarxa, però en el moment en què la troba, algun dels processos que fa demanen una quantitat d'energia que l'Arduino no és capaç de lliurar.

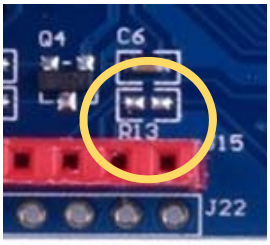
A la il·lustració 23 es mostra com l'escut disposa de dos *jumpers* (marcats com a J11 i J12) que permeten configurar si es farà una comunicació serial els pins D0 i D1 o bé a través dels ports D7 i D8. Tal i com s'ha explicant anteriorment, l'Arduino té ocupats els pins D0 i D1 per a comunicacions internes, motiu pel qual s'ha triar utilitzar els pins D7 per a la transmissió i D8 per a la recepció.



Il·lustració 23 Detall jumpers mòdul GSM

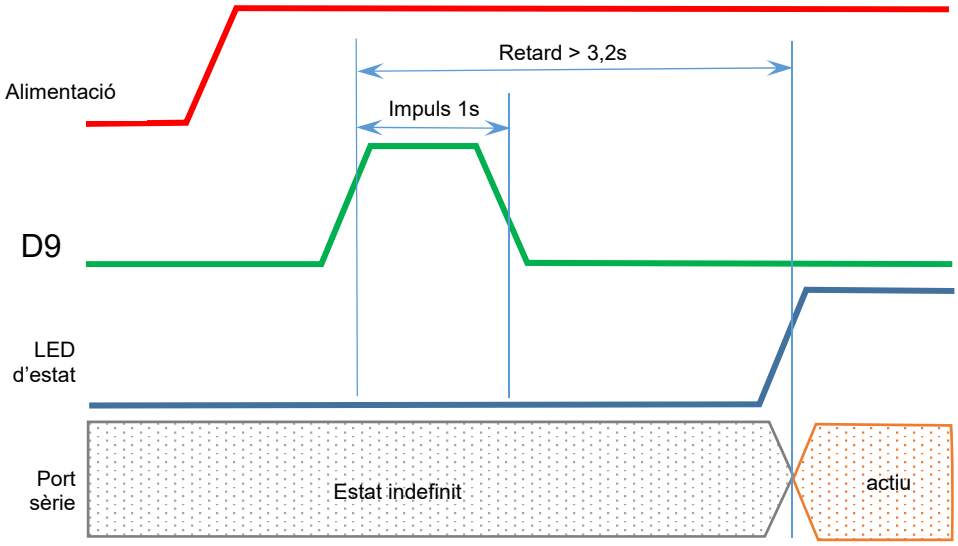
D'altre banda, per a l'encesa del mòdul hi ha dues opcions: pitjar un micro polsador durant dos segons o bé fer-ho a través d'un pin. Donat que es vol que el sistema sigui autònom, aquest segon mètode és l'escollit. Aquesta opció obliga a fer una soldadura sobre la resistència

R13 que le indica a la placa que l'activació es fa mitjançant un ordre lògica des dels pin corresponent i no des del mini polsador que disposa el mòdul.



Il·lustració 24 Detall soldadura R13 mòdul GSM

L'escut no dóna opcions sobre quin pin ha de ser qui gestiona l'encesa del mòdul i obliga que aquest sigui el D9. Cal tenir present que per a l'activació no hi ha prou amb només activar el pin sinó que s'ha de seguir una seqüència:



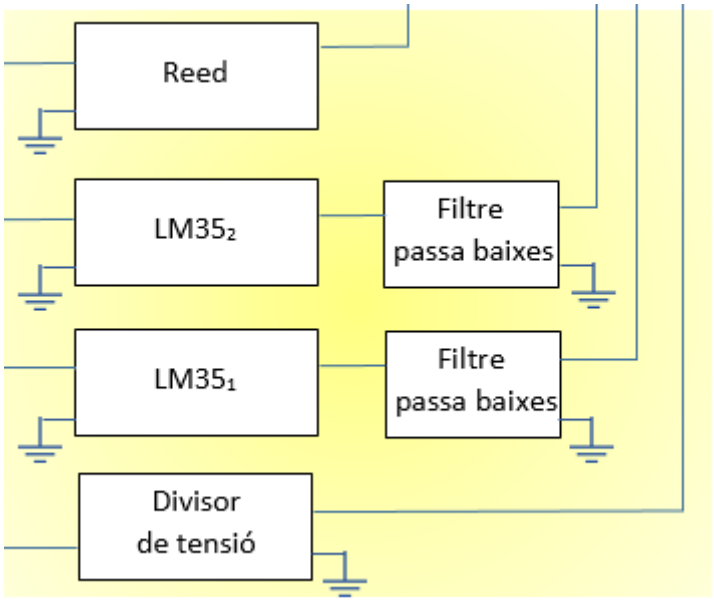
Il·lustració 25 Seqüència d'encesa del mòdul GSM per software

És a dir, per activar el mòdul cal mantenir actiu el pin D9 durant 1 segon i esperar com a mínim 2,2 segons més per tal que comenci la transmissió de dades a través del port sèrie.

2.3.6 LES CONNEXIONS DELS SENSORS

Els principals sensors que formen part del sistema són:

- Un sensor *reed*, que permet saber si la comporta de l'aire condicionat està oberta o tancada i per tant, de manera indirecta, si l'aparell de climatització està o no en funcionament
- Un sensor de temperatura que permet conèixer a quina temperatura està enviant l'aire fred a la sala l'aire condicionat.
- Un altre sensor de la temperatura que permet conèixer la temperatura de la sala on es troben els servidors.
- Un divisor de tensió que permet conèixer la tensió a la que es troben les bateries i d'aquesta forma poder determinar si cal carregar-les o no.

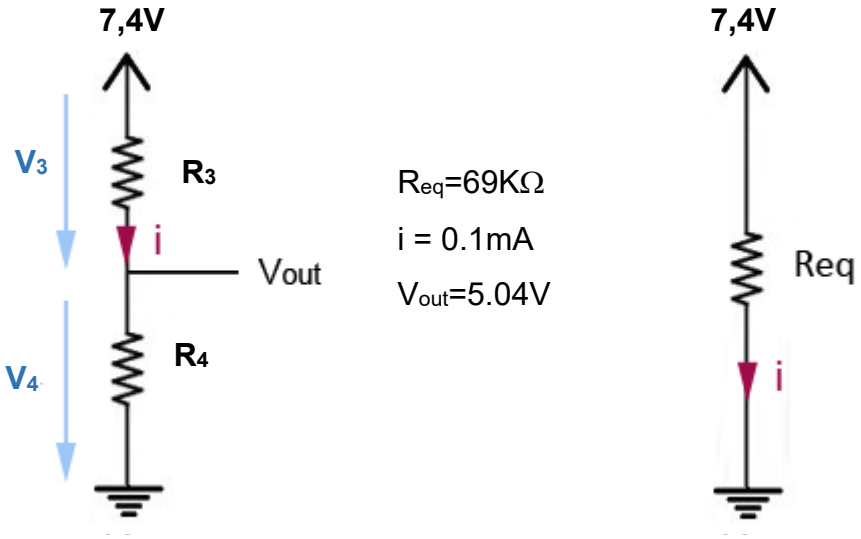


Il·lustració 26 Esquema dels principals sensors

Donat que les bateries tenen una tensió nominal de 7,4V i que aquesta tensió és massa elevada per les entrades de l'Arduino, ha calgut fer un divisor de tensió per permet saber de manera indirecta la tensió a les bateries en funció del valor llegit al divisor de tensió.

Aprofitant la calculadora de divisors de tensió que podem trobar al portal d'en Luis Lamas (LUIS LLAMAS, 2017), els valors de les resistències que formen el divisor de tensió són R_3=22KΩ i R_4=47KΩ. S'ha triat uns

valors elevats de les resistències per tal de minimitzar el corrent elèctric emprat per conèixer la tensió i d'aquesta manera tenir el mínim impacte possible en el consum de l'energia acumulada a les bateries.



Il·lustració 27 Divisor de tensió

Així doncs, la lectura que tindrem al pin corresponent al divisor de tensió serà un 68% del valor de la tensió de les bateries.

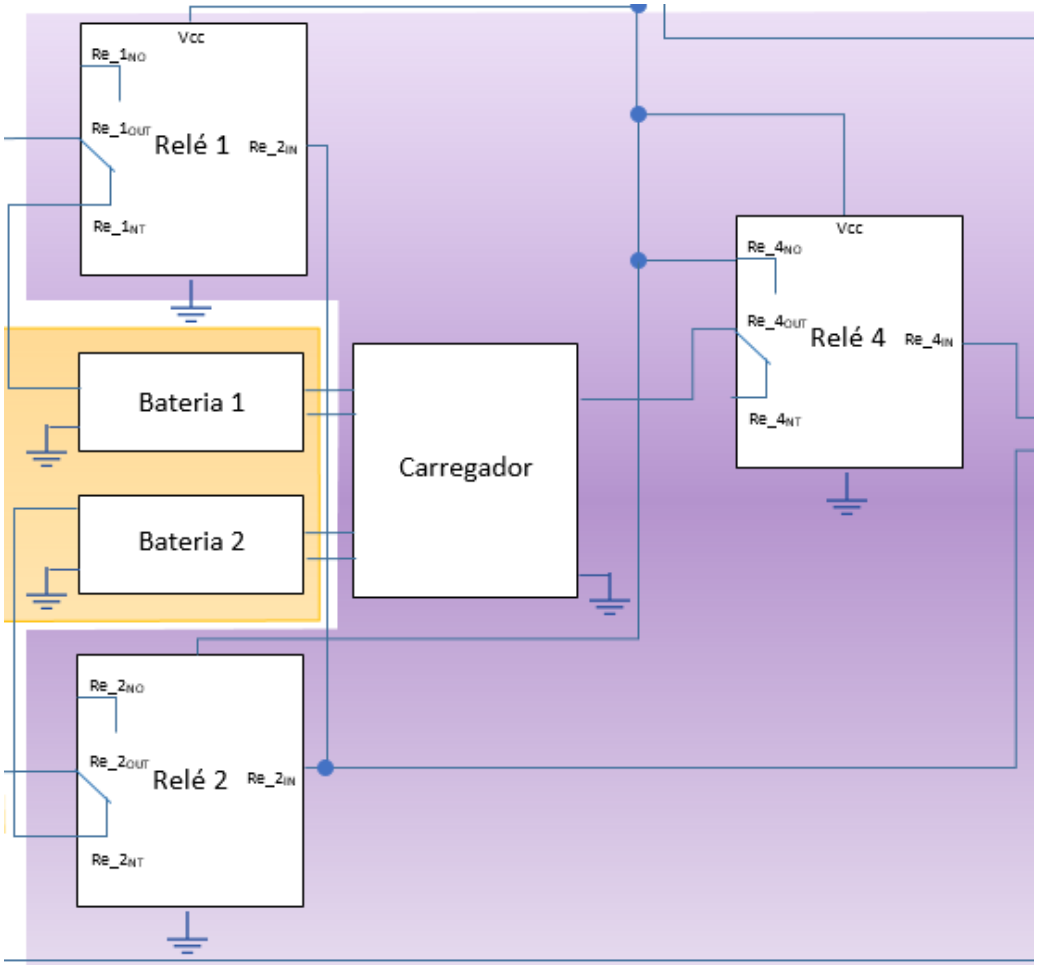
$$V_{out} = \frac{R_4}{R_3 + R_4} V = \frac{47K\Omega}{69K\Omega} V = 0{,}68V$$

2.3.7 El circuit de càrrega de les bateries

La càrrega de les bateries es gestiona a través de 3 relés, tal i com es pot veure a la il·lustració següent. El relé 4 és l'encarregat de fer d'interruptor del carregador, donant-li els 5V que necessita directament des de la font d'alimentació provinent de la xarxa elèctrica. Per connectar el carregador a l'alimentació caldrà activar el relé 4 posant la seva entrada Re_4_{IN} a 0V.

Abans però, per tal de minimitzar possibles problemes durant els períodes de càrrega, caldrà desconnectar les bateries del circuit d'alimentació i també entre elles (cal recordar que estan connectades en

paral·lel i que tant s'alimenten entre elles). Així doncs, abans d'activar el relé 4 del carregador caldrà activar els relés 1 i 2 per tal de desconnectar les bateries.



Il·lustració 28 Circuit de càrrega de les bateries

Per tornar a connectar les bateries al circuit caldrà fer el procés contrari. Primer desactivar el carregador i posteriorment connectar les bateries

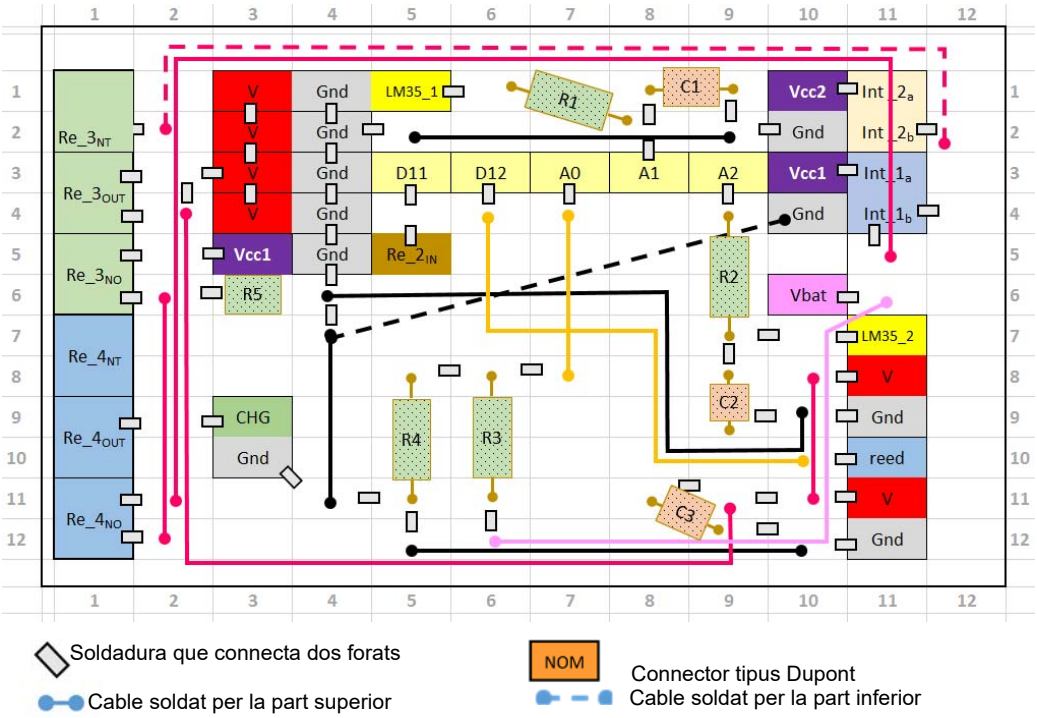
2.3.8 La placa de connexions

En el moment de d'abandonar el temps de desenvolupament amb les connexions mitjançant les plaques protoboard i voler passar a un prototip comercial, va aparèixer la necessitat de crear una placa on fer totes les connexions.

El disseny d'aquesta placa s'ha fet seguint les següents premises:

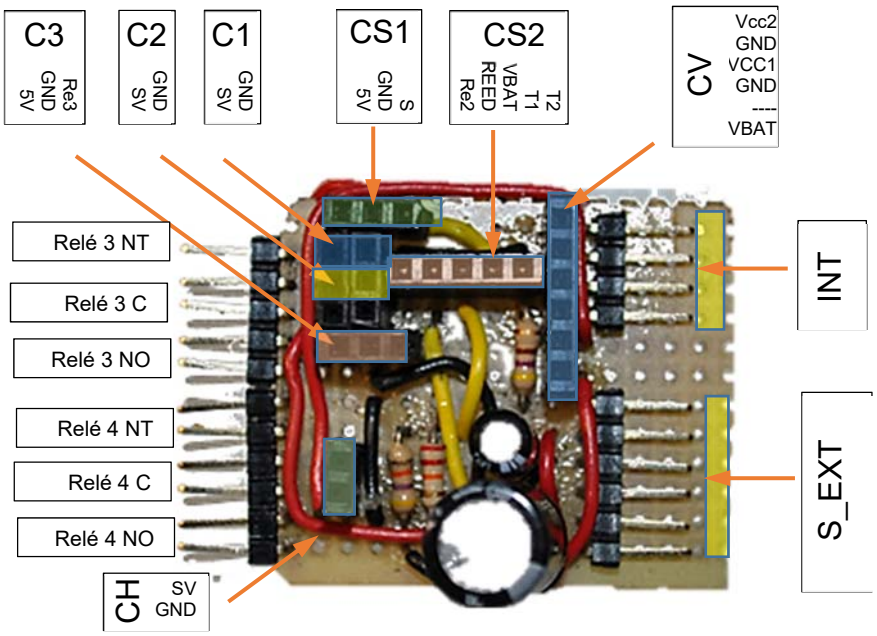
- utilitzar un espai reduït per tal de poder-la encabir a la caixa escollida pel sistema
- facilitar que el sistema sigui muntable i desmuntable per facilitat la substitució de mòduls o el canvi de bateries
- Agrupar les connexions de tal manera que sigui fàcil desconnectar els cables i tornar-los a connectar a la seva posició

Per fer el disseny va ser molt útil la utilització d'un full de càlcul del Microsft Excel on cada casella representa un forat de soldadura de la placa de soldadures. La següent il·lustració mostra al disseny compacte que s'ha implementat.



Il·lustració 29 Detall contactes placa connexions

Aquest disseny sobre el paper passat a la realitat ha quedat com es veu a la següent il·lustració.



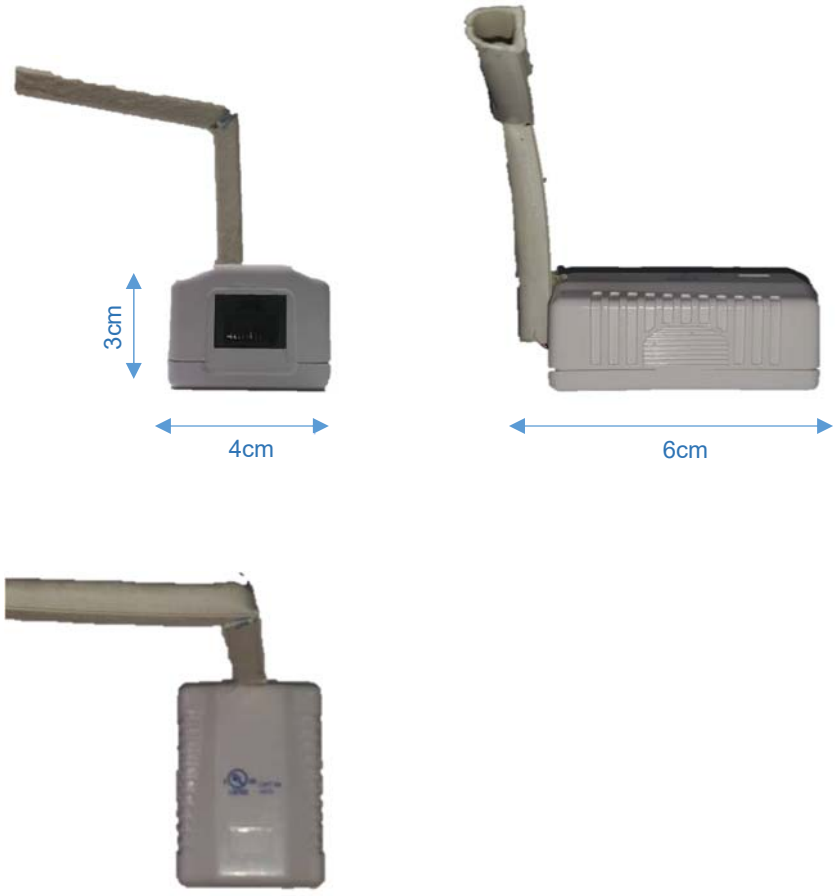
Il·lustració 30 Detall de la placa de connexions

On:

- CS1 ⇒ Connector de sensors que va directament al sensor de temperatura de la sala que es troba a una de les vores de la caixa.
- CS2 ⇒ Connector de sensors que connectar els pins de l'Arduino amb aquesta placa.
- C1 ⇒ Connector d'alimentació per alimentar el mòdul GSM i el propi Arduino.
- C2 ⇒ Connector d'alimentació per alimentar el mòdul dels relés 1 i 2, que són els encarregats de obrir la connexió de les bateries amb el circuit mentre dura la càrrega de les mateixes.
- C3 ⇒ Connector d'alimentació i control per alimentar el mòdul dels relés 3 i 4 i controlar el relé 4, encarregar d'activar el carregador de les bateries.
- CH ⇒ Connector d'alimentació que per alimentar el carregador de les bateries. Hi ha tensió quan està activat el relé 4
- INT ⇒ Connector a l'interruptor bipolar utilitzar per desactivar tota l'alimentació del sistema.
- S_EXT ⇒ Connector als sensors externs al quals s'arriba mitjançant un cable de xarxa.

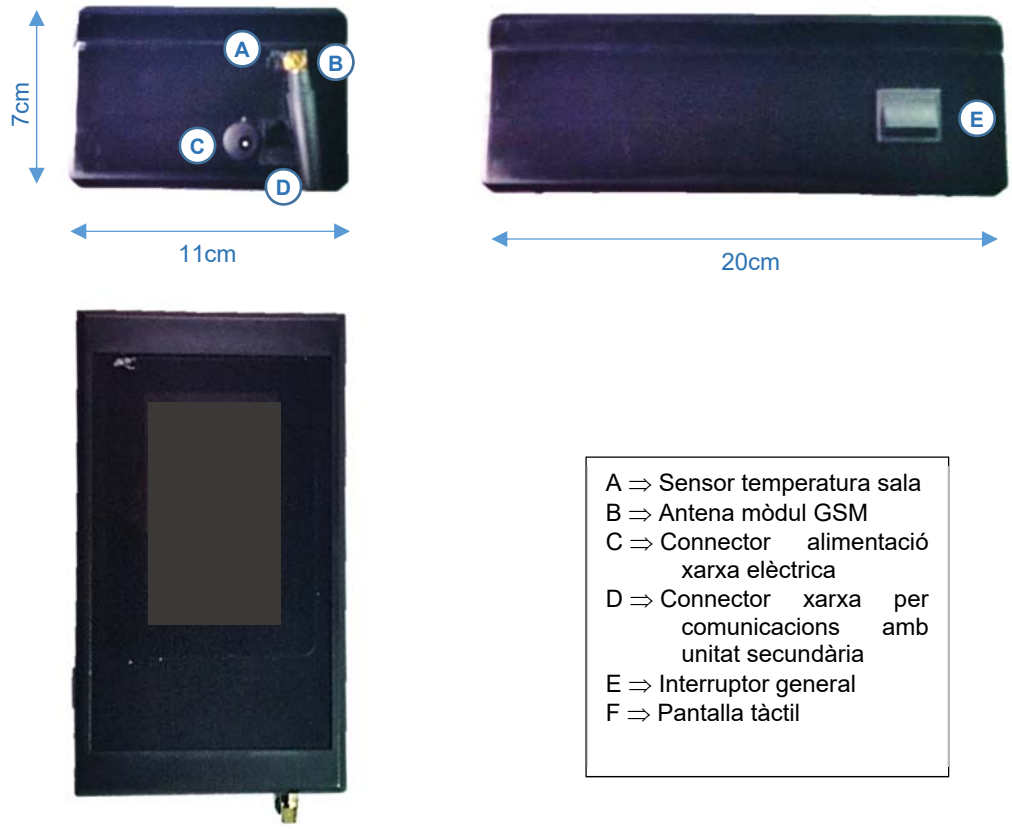
2.3.9 L'ASPECTE FINAL

La unitat secundària encarregada de encabir el sensor reed i el sensor de temperatura de la sortida d'aire de l'aire condicionat, juntament amb la connexió de xarxa ha quedat amb el següent aspecte:



Il·lustració 31 Aspecte de la unitat secundària

La unitat principal, amb la pantalla tàctil i tota la resta de maquinari al seu interior té l'aspecte que es veu a continuació:



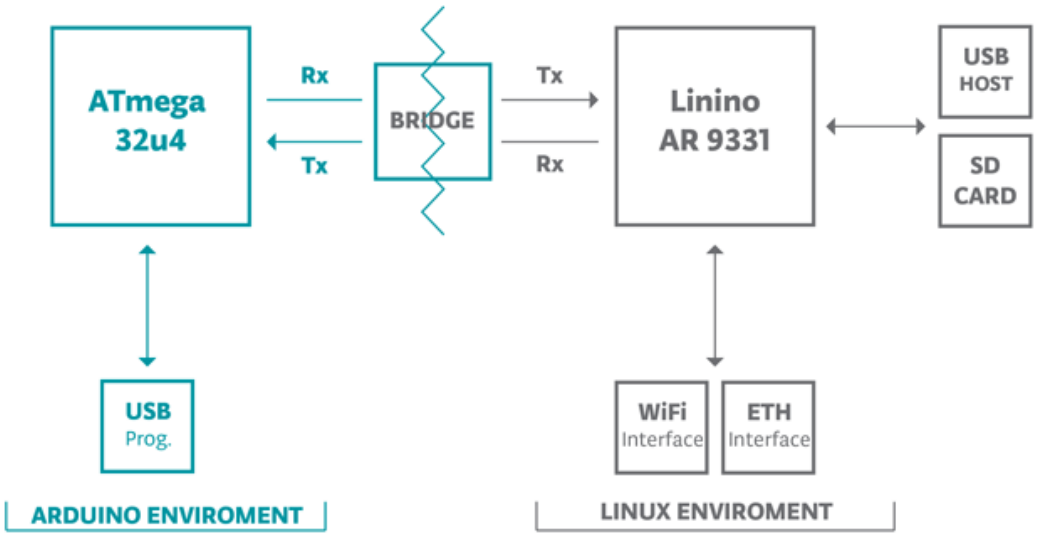
Il·lustració 32 Aspecte de la unitat principal

2.4 El programari i la programació

Un cop descrits els principals aspectes relacionats amb el dispositius físics i les seves interconnexions, a partir d'aquest punt s'explicarà els aspectes relacionats amb el programari utilitzat i allò que s'ha programat.

2.4.1 L'INTERIOR DE L'ARDUINO YÚN

Fins el moment s'ha fet referència a la part elèctrica del l'Arduino que és més o més comú a la resta de microcontroladors de la plataforma Arduino, com per exemple l'Arduino UNO. A partir d'ara cal parlar d'una altra part de la placa que és el que realment la diferencia.



Il·lustració 33 Les dues parts de l'Arduino Yún

A la figura anterior s'il·lustra aquest fet. Realment la placa Yún està formada per una part que és en microcontrolador –és el que té de comú amb la resta de plaques de la plataforma- i una segona part que és un microprocessador amb connexions Ethernet i Wi-Fi, emmagatzematge amb una targeta SD i un port *host* USB que permet la connexió d'altres dispositius.

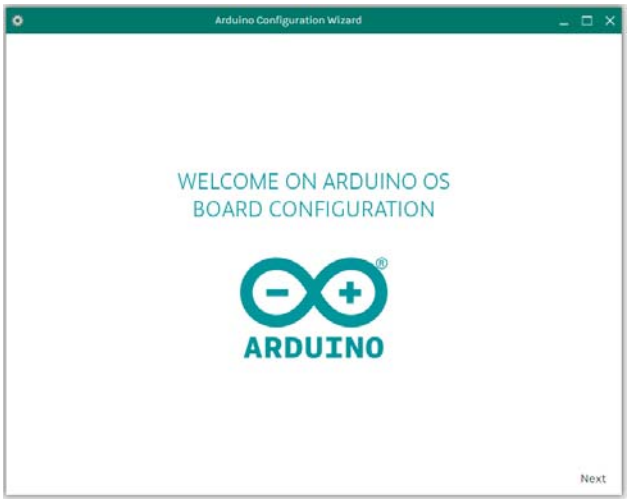
El sistema operatiu utilitzat per aquest microprocessador és una distribució de Linux anomenada "Linino" que es basa en el projecte OpenWrt (OpenWrt, 2017). A aquest projecte se'l descriu com la distribució Linux per a dispositius embeguts.

El projecte OpenWrt pretén, en comptes de disposar d'un únic firmware que es pugui compartir per molts dispositius, disposar d'un sistema d'arxius que siguin compatibles entre ells i que permetin , triant-ne uns o uns altres, disposar d'un sistema operatiu funcional per a la major part de dispositius.

2.4.2 LA PRIMERA CONFIGURACIÓ

Així que es connecta l'Arduino Yún, un cop arrencat el sistema –per tal d'haver completat el procés ha de passar més d'un minut- apareix una xarxa sense cables Ad-Hoc amb un nom que segueix el format ArduinoYun- XXXXXXXXX, on cal substituir aquest conjunt de lletres X per l'adreça MAC del dispositiu.

Es tracta d'una Wi-Fi oberta a la qual es pot connectar des de qualsevol dispositiu que tingui una connexió Wi-Fi. Les plaques Arduino Yún venen configurades amb la IP 192.168.240.1 i màscara 255.255.255.0. De totes maneres no cal saber això ja que la plana de benvinguda apareix automàticament.



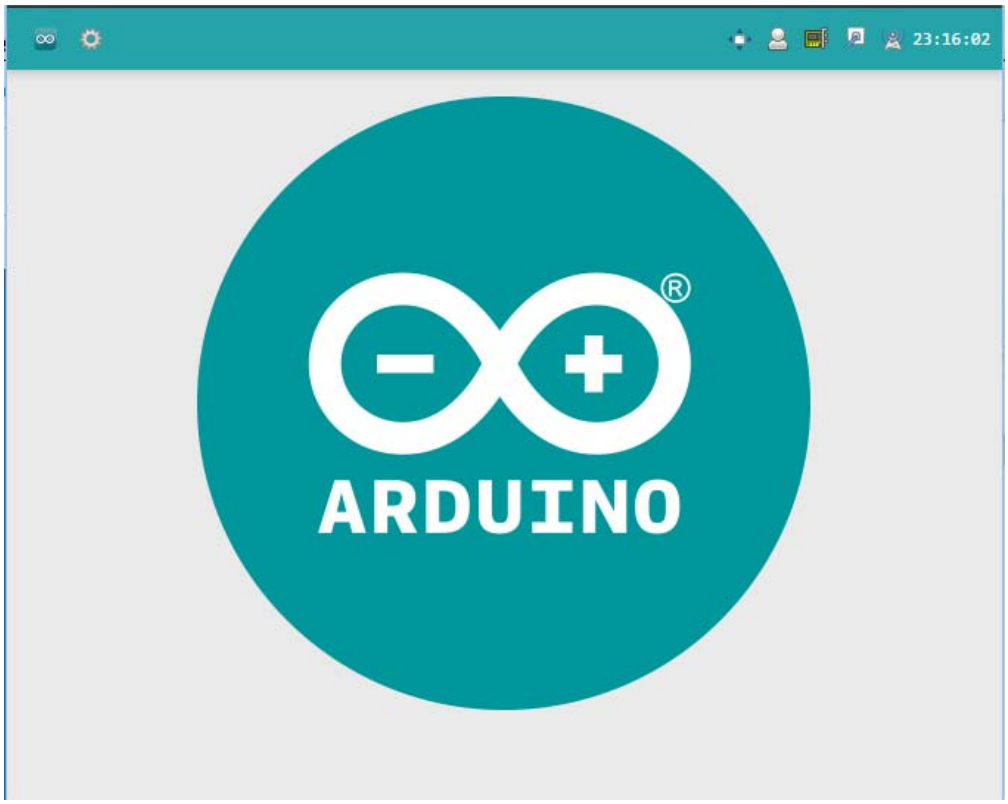
Il·lustració 34 Pàgina de benvinguda i configuració

Un cop acabada aquesta configuració inicial s'haurà donat un nom al dispositiu, s'haurà creat una paraula de pas per tal d'accedir a l'administració de la placa i s'haurà configurat l'accés a una Wi-Fi existent.

A l'annex 6.1 es poden veure les captures del procés. En acabar el procés es produirà un reinici del sistema que farà que s'hagi perdut la connexió i caldrà – si més no en sistemes Microsoft Windows- trobar la IP que la xarxa Wi-Fi li ha assignat a l'Arduino per DHCP.

2.4.3 L'ACTUALITZACIÓ DEL SISTEMA

Escrivint l'adreça obtinguda en un navegador web, s'arribarà a un primer portal d'administració de la placa –Això sí, caldrà haver escrit prèviament la clau la clau introduïda en el pas anterior-. L'aspecte de l'entorn serà semblant a la següent figura:

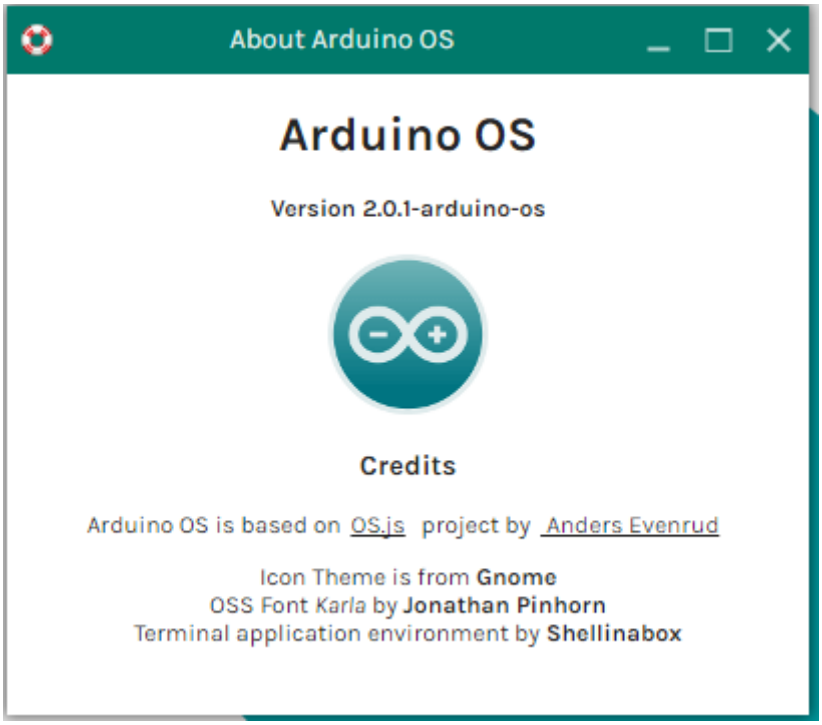


Il·lustració 35 Panel d'administració Arduino OS v2.01

Aquesta versió ja està obsoleta i de seguida el sistema avisa que hi ha una nova versió. Caldrà doncs fer una actualització del sistema. Per això anirem a la primera icona de la part superior esquerra i triarem l'opció *Arduino Package Manager*.

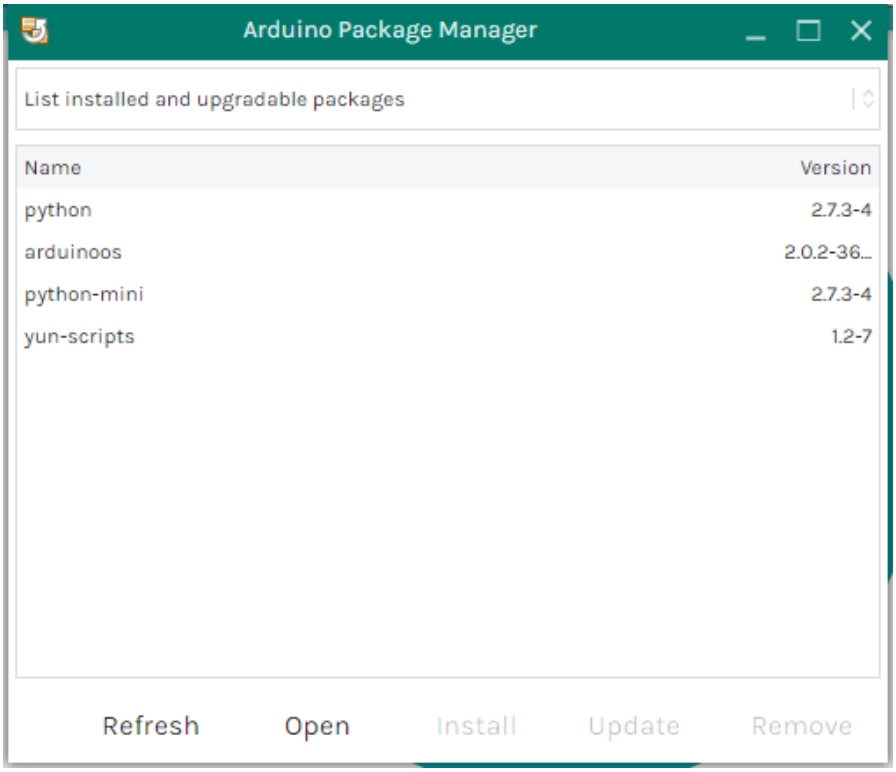
Il·lustració 36 Menú principal Arduino OS v2.0.1

Si abans de fer l'actualització mirem la informació de la versió incial veurem que es tracta de la versió 2.0.1 de l'Arduino OS.



Il·lustració 37 Versió inicial del sistema operatiu

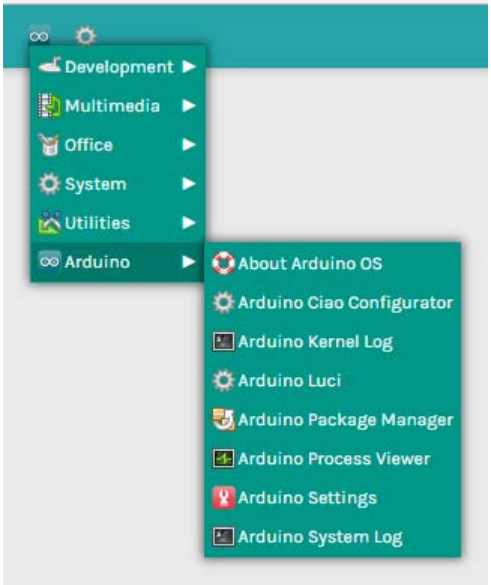
Obrint l'Arduino Package Manager es comprovarà lo fàcil que és complementar la instal·lació bàsica amb la que arriba la placa. En comptes de triar el llistat de paquets disponibles canviarem la selecció al llistat de paquets instal·lats i actualitzables.



Il·lustració 38 Actualitzacions del sistema operatiu

L'experiència viscuda per l'autor d'aquest treball recomana no fer totes les actualitzacions de cop donat que en funció de la combinació de l'odre d'actualitzacions triada poden aparèixer problemes que deixen el sistema de forma inestable. Primer de tot millor actualitzar el sistema arduinoos a la nova versió proposada i després ja es faran la resta d'actualitzacions.

Finalitzada l'actualització del sistema operatiu, s'estarà treballant amb la versió 2.0.3 del sistema operatiu. El canvi es notarà de seguida ja que el menú que apareixia en la versió anterior i que es pot veure a la il·lustració 36 apareix organitzat d'una forma molt diferent.



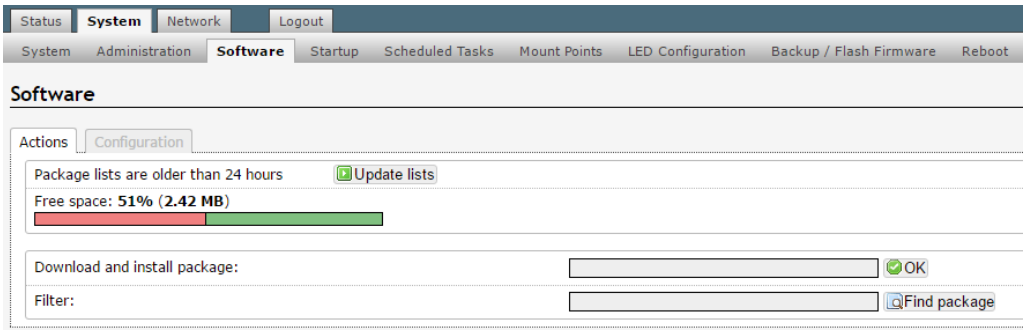
Il·lustració 39 Nou menú principal Arduino OS v2.0.3

A vegades apareixen problemes deguts a problemes de refresc de la memòria cau del navegador. En aquest casos val la pena canviar de navegador o bé obrir una finestra d'incògnit per tal de forçar el refresc dels arxius.

2.4.4 L'ARDUINO LUCI

Per disposar de la màxima informació sobre l'estat de la part linux de la placa Yún el millor és accedir al portal anomenat Luci. Si s'accedeix des de la plana inicial aquest portal apareix com a una aplicació embeguda dins d'ella. Però també es pot accedir al portal escrivint com a URL la IP de la placa seguida de /cgi-bin/luci .

Una mostra de les molta informació que ofereix aquest portal d'administració del linino es veu en la quantitat de pestanyes que apareixen a la captura següent:



Il·lustració 40 Algunes pestanyes del portal LuCI

En el proper apartat es comentarà la importància de la informació que apareix en aquesta il·lustració.

2.4.5 Instal·lant la targeta de memòria SD

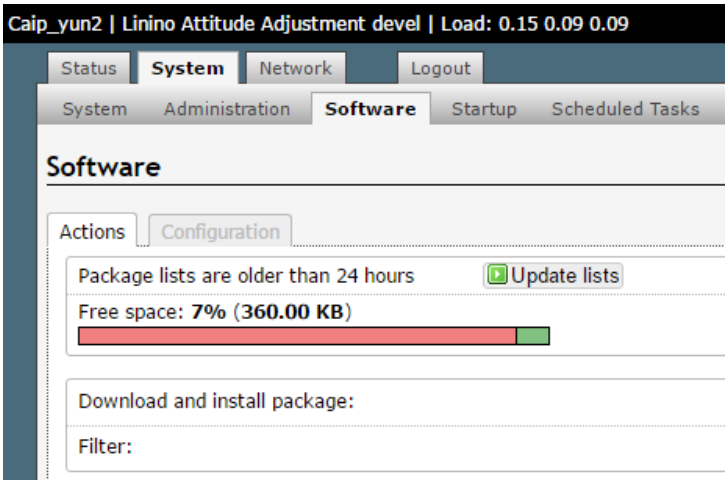
En el treball que ens ocupa necessitem accedir a la targeta SD que la placa incorpora. Per tal de poder-la utilitzar i treure-li el màxim de profit, hi ha un *sketch* en el portal d'Arduino anomenat YunDiskSpaceExpander (Arduino, 2017) que permet particionar la targeta SD de forma adient per a ser utilitzada en l'Arduino Yún.

Aquest *sketch* bàsicament executa una sèrie de comandes Linux encarregades de particionar i formatar la targeta. El procés crea una partició FAT32 (que per tant estarà limitada als 4GB de capacitat) i una altra que estarà formatada en el sistema d'arxius EXT propi de Linux.

Un cop creades les particions, el linino utilitzarà aquesta partició Linux com a una extensió de la seva de la seva capacitat d'emmagatzematge. L'extensió FAT32 servirà per a deixar dades i per a compartir informació amb sistemes Windows. La partició Linux ocuparà tota la part de la targeta que no ocupi la partició Windows.

És important tenir com a mínim 1MB d'espai a la memòria flash de l'Arduino per tal que *sketch* funcioni correctament. Es dóna la circumstància que si s'han fet les 4 actualitzacions de programari

proposades pel sistema –veure apartat anterior-, l'espai disponible a la memòria flash és inferior al megabyte d'espai lliure que requereix *l'sketch* per tal de poder fer les noves particions a la targeta.

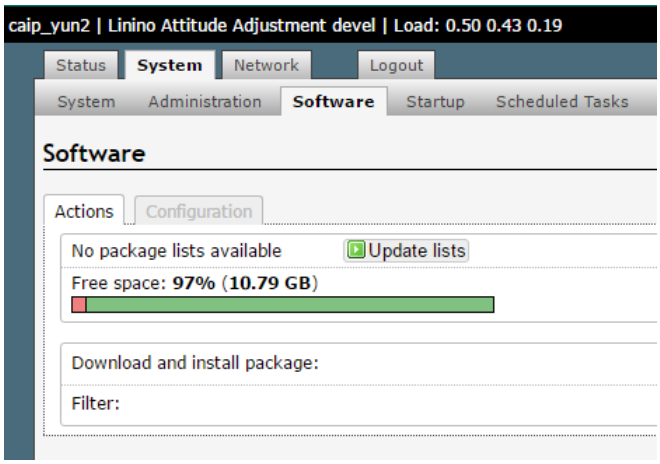


Il·lustració 41 Espai disponible al sistema després d'actualitzar-ho tot

Així doncs hi ha dos opcions:

- Si s'han fet les actualitzacions abans indicades ⇒ cal des-instal·lar alguna aplicació per tal d'alliberar recursos o bé restaurar el sistema a les opcions per defecte i tornar a començar el procés.
- Si encara no s'han fet les actualitzacions ⇒ esperar a fer-les un cop feta la formatació de la targeta ja que aleshores hi haurà molt més espai disponible.

En aquest treball s'ha utilitzat una targeta SD de 16 GB, dels que 3,7GB s'han dedicat a la partició FAT32. Tornant a consultar l'espai disponible després de la formatació de la targeta SD es pot comprovar com ara es disposa de molt més espai per a instal·lar programes.



Il·lustració 42 Espai del sistema disponible post instal·lació SD

A l'annex 6.2 es mostren les captures del procés de formatació mitjançant l'*sketch* YunDiskSpaceExpander

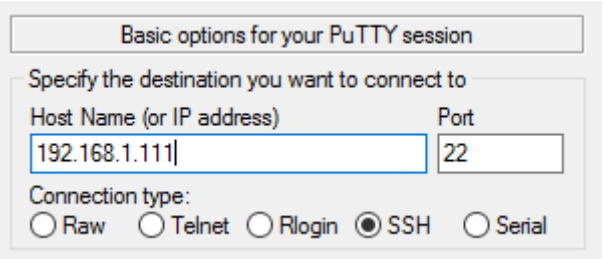
2.4.6 Accedint en mode consola al sistema operatiu Linino: l'aplicació PuTTY

Hi ha un programa força conegut per els usuaris de sistemes basats en Linux per accedir remotament als equips. Es tracta d'un programa que mitjançant diferents protocols permet accedir a les consoles dels equips remots i interactuar amb ells a partir de l'escriptura de comandes.

Aquest programa és diu PuTTY i és Open Source, per la qual cosa es pot utilitzar sense problemes i es pot descarregar des de la seva web (Putty.org, 2017).

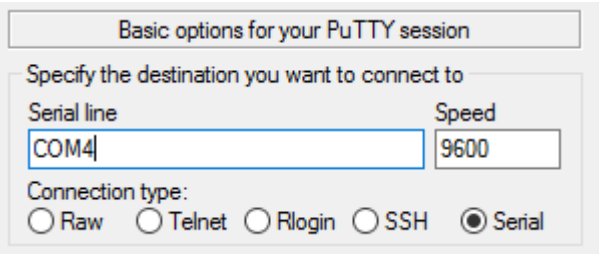
En el cas que es ocupa podem accedir a l'Arduino Yún amb dos protocols diferents:

- Si coneixem la IP de dispositiu ⇒ podem accedir mitjançant un connexió SSH.



Il·lustració 43 Dades connexió SSH mitjançat PuTTY

- Si tenim una connexió USB amb el dispositiu ⇒ podem accedir mitjançant un connexió serial. En aquest cas caldrà saber quin port COM és el que té assignat l'Arduino. Com a qualsevol comunicació sèrie, cal saber la velocitat de connexió, el nombre de bits d'informació, el nombre de bits de stop i la paritat. Per establir la connexió amb l'Arduino els valors que hem de posar respectivament són 9600,8,1 i cap paritat



Il·lustració 44 Dades connexió serial mitjançant PuTTY

En qualsevol dels dos casos demana usuari i clau per accedir a la consola. L'usuari és root i la clau, la escollida en el moment de la primera configuració.

Poca cosa es podrà fer amb la consola si no es coneixen les comandes a executar, però Internet és ple d'informació i es poden provar moltes comandes per obtenir informació dels sistema.

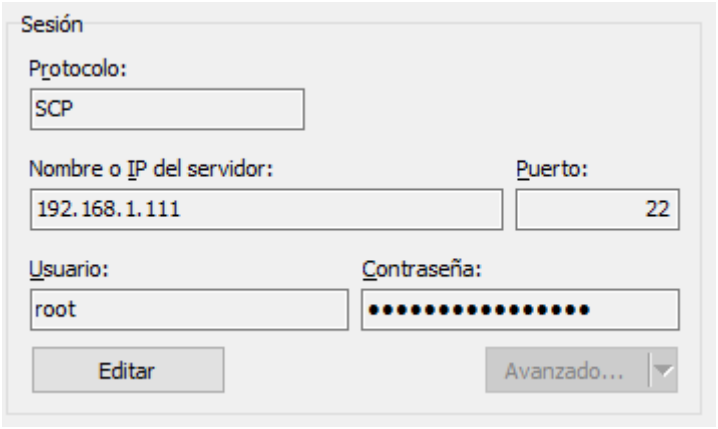
Alguns exemples de comandes que s'han utilitzat durant l'elaboració del treball han estat:

- `ifconfig wlan0` ⇒ aquesta comanda retorna informació sobre *l'interface* Wi-Fi dels dispositiu, amb la qual cosa permet saber la IP assignada al mateix

- ifconfig wlan0 |awk '/addr:/ {print $2}' ⇒ aquesta és la mateixa que l'anterior però filtra la informació retornada i només retorna el que hi ha a la segona posició de la línia on s'ha trobat la paraula cercada "addr:". En aquest cas obtenim directament la IP. Si en comptes de *print $2* s'escriu *print $4*, el que s'obté és la màscara, que és l'element que apareix en quarta posició.
- /usr/bin/pretty-wifi-info.lua | grep 'IP address:' ⇒ aquesta és una altra comanda per poder obtenir la IP i la màscara de l'interface Wi-Fi.
- date "%Y-%m-%d %H:%M:%S" ' ⇒ aquesta comanda retorna la data i hora actuals del sistema.
- mv –f rutaOrigen rutaDesti' ⇒ aquesta comanda renombra l'arxiu rutaOrigen com a rutaDesti, sobreescrivint el destí en cas que ja existeix i sempre que la data de l'arxiu existent sigui anterior a l'arxiu que s'està renombrant

2.4.7 Accedint en mode gràfic al operatiu linino: l'aplicació WinSCP

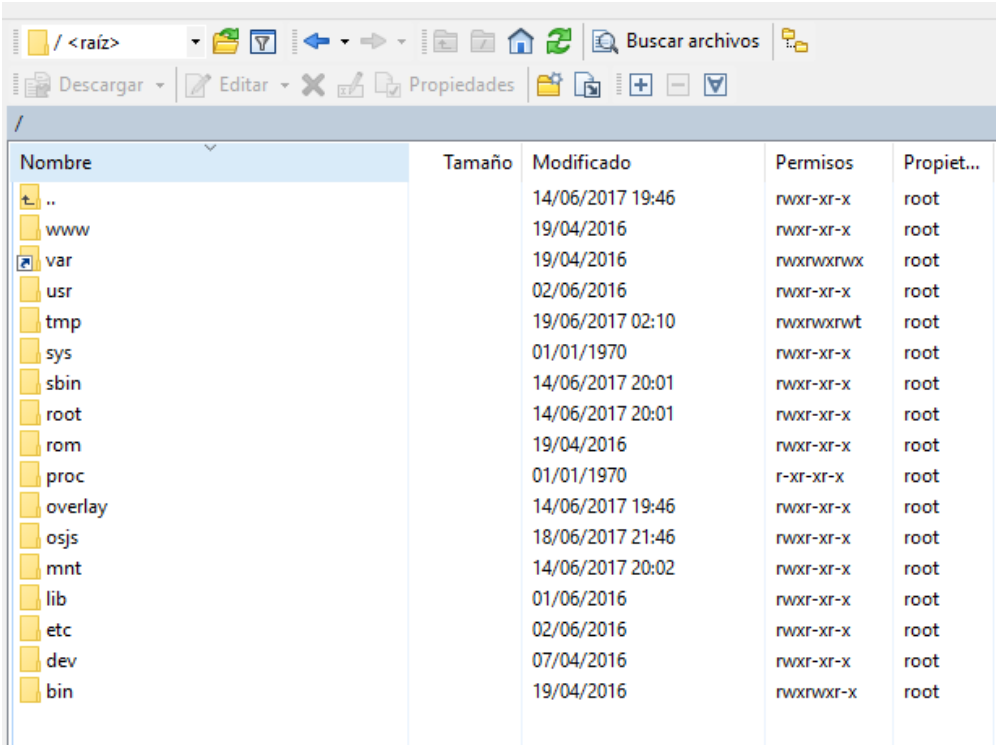
Es tracta d'una aplicació de programari lliure que permet accedir de forma gràfica a un sistema remot per tal, sobretot, d'intercanviar arxius. Per accedir-hi només cal saber la IP del sistema al que es vol accedir, l'usuari i la clau.



Il·lustració 45 Dades connexió amb WinSCP

Un cop establerta la connexió es visualitzen les carpetes i els arxius dels sistemes remot i local i és fàcil editar i moure arxius d'un lloc cap a l'altre i a l'inrevés.

L'autor ha utilitzat aquesta aplicació també per crear accessos directes que, col·locats a la carpeta adient, permeten facilitar l'accés web a les carpetes del projecte.



Il·lustració 46 Directori arrel de la distribució linino

L'aplicació WinSCP es pot descarregar des del seu propi portal (WinSCP, 2017) que apareix a la bibliografia.

2.4.8 Instal·lant PHP en el servidor web de l'Arduino

El linino ja porta instal·lat un servidor de planes web. Aquest servidor, però, només és capaç de servir planes senzilles tipus HTML, i els seus arxius associats com són el arxius d'estil –amb extensió *css*-, arxius de scripts escrits en javascript –amb extensió *js*-, arxius de imatges –amb extensions *jpg, gif, png, bmp*-, arxius de text –amb extensió *txt*-, i no gaires tipus més. Amb això realment ja es poden fer moltes coses, però les possibilitats es multipliquen quan s'instal·la un intèrpret de llenguatges de tipus servidor com pot ser el PHP.

Per instal·lar aquest intèrpret cal executar unes comandes i el més fàcil és fer-ho de d'una connexió a la consola del linino. La recomanació de l'autor del treball és obrir una sessió amb l'aplicació PuTTY i executar les següents comandes que expliquen a l'entrada de blog de Antony García González en el portal panamaHitek (Instalación y configuración de PHP en el Arduino Yún, 2017)

Es tracta d'executar unes comandes d'instal·lació de paquets del php 5:

```
opkg update
opkg install php5
opkg install php5-mod-json
opkg install php5-mod-cURL
opkg install php5-cli
opkg install php5-cgi
```

I després unes comandes per modificar la configuració del servidor de planes web de que ve per defecte amb linino i que s'anomena **uhttpd**

```
uci set uhttpd.main.interpreter=".php=/usr/bin/php-cgi"
uci set uhttpd.main.index_page="index.html index.htm default.html default.htm
      index.php"
uci commit uhttpd
sed -i 's,doc_root.*,doc_root = "",g' /etc/php.ini
sed -i 's,;short_open_tag = Off,short_open_tag = On,g' /etc/php.ini
```

I finalment reiniciar el servidor

```
/etc/init.d/uhttpd restart
```

2.4.9 Programant planes web que interactuen amb l'arduino

Tal i com s'ha comentat en l'apartat 2.4.1, dins l'Arduino Yún hi conviuen dos mons independents que es comuniquen mitjançant un interlocutor.

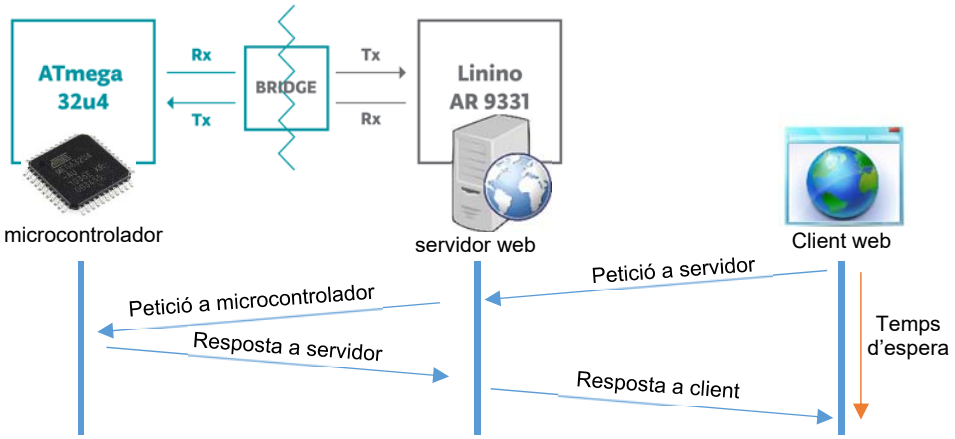
Aquest interlocutor s'anomena Bridge i s'encarrega de fer les gestions per tal que la comunicació flueixi entre els dos interlocutors. Gràcies a ell, des del servidor web es poden consultar i modificar les entrades i sortides del microcontrolador.

De tot això se n'encarreguen 3 llibreries que cal incloure a l'*sketch* i de les qual ja en parlarem en l'apartat següent.

Des del punt de vista web, l'únic que cal saber és que hi ha unes adreces web que es poden consultar i que o bé retornen una informació com a resposta a la nostra consulta o bé agafen informació de la URL que hem enviat com a consulta i fan amb ells el que els calgui.

Així doncs, des del punt de vista web, només es traca d'escriure les URL adequades i de la resta ja se'n encarregarà el Bridge.

Cal tenir present que les peticions fetes a aquest interlocutor no poden ser gaire complexes ja que el microcontrolador que hi ha a l'altre banda té unes capacitats limitades.



Il·lustració 47 Esquema de funcionament petició web

Com es pot veure a la il·lustració 47, des que el navegador de l'usuari fa una petició al servidor web fins que li arriba la resposta passa un cert interval de temps. Aquest temps es veu incrementat respecte al que triga un servidor normal pel fet que aquest ha d'interrogar al seu torn al microcontrolador per que li doni resposta a la seva consulta.

Per a les peticions al servidor s'ha utilitzat la tecnologia AJAX que permet actualitzar parts de la plana que s'està visualitzant sense necessitat de recarregar-la sencera. Aquest fet ajuda a descarregar de tasques a fer

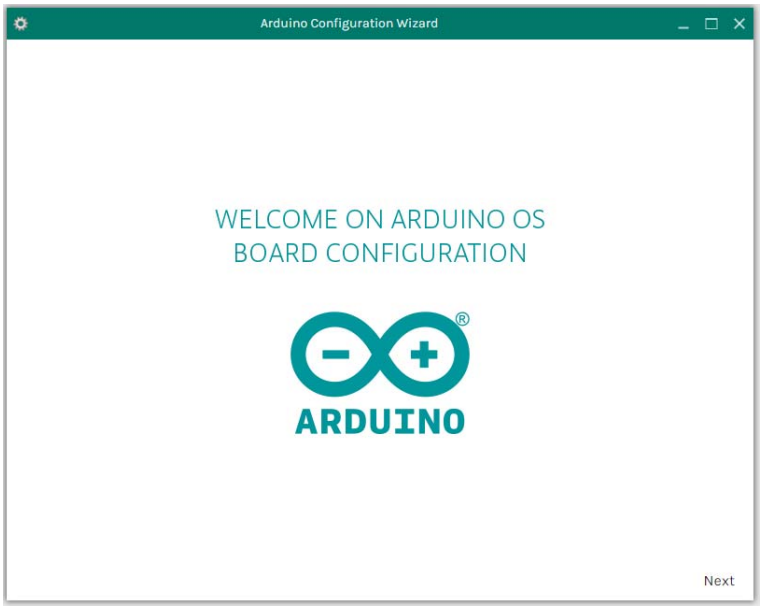
al servidor donat no li cal reenviar al client informació de la qual ja disposa.

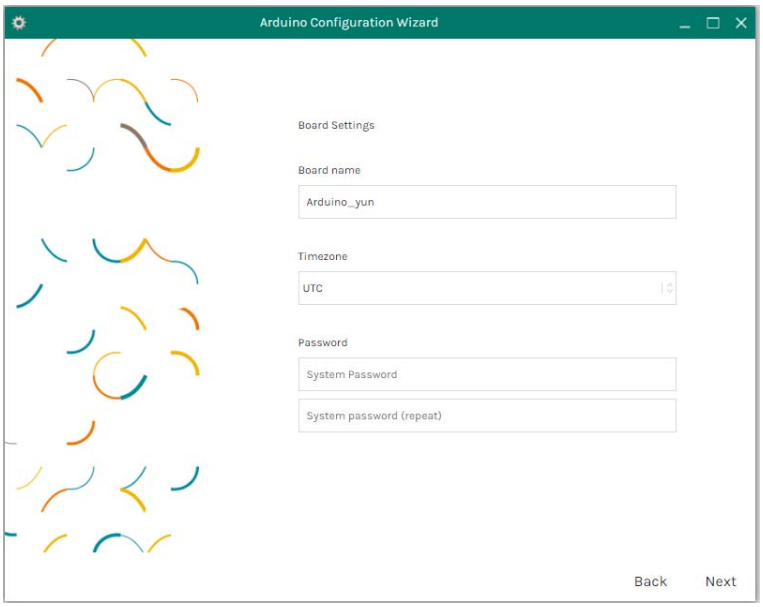
Per accedir al portal de visualització de les dades i configuració de les opcions cal teclejar la següent URL:

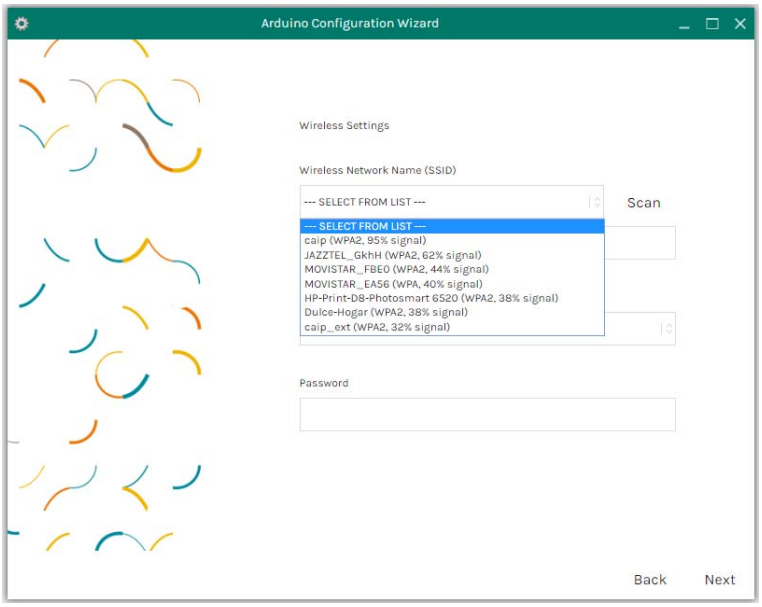
http://adreça-del-yun/tfg

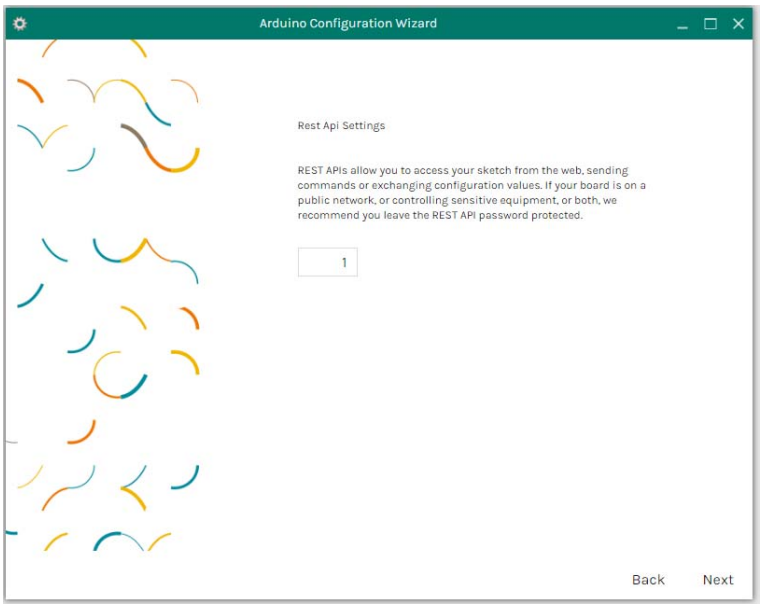
on *adreça-del-yun* cal substituir-ho per l'adreça IP del dispositiu.

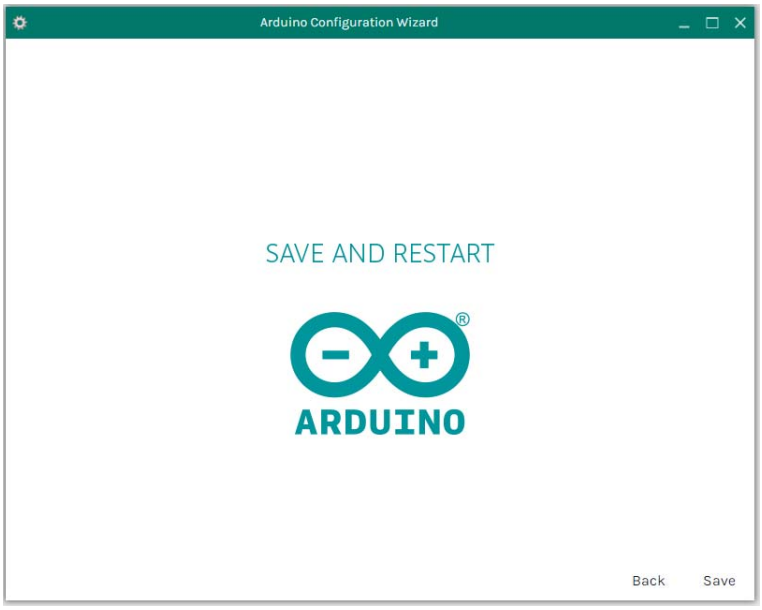
Per aconseguir aquest comportament -que no és el que fa l'Arduino per defecte- cal crear un vincle a la carpeta arrel del servidor amb el nom desitjat –en el meu cas, tfg-. A l'
annex 6.1 captures del procés de configuració inicial de l'arduino yún




Arduino Configuration Wizard
Board Settings
Board name
Arduino_yun
Timezone
UTC
Password
System Password
System password (repeat)
Back
Next

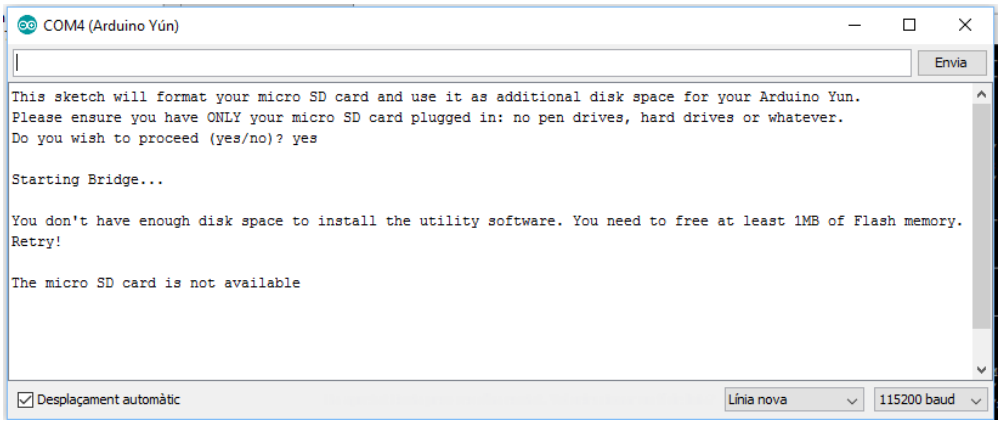

Arduino Configuration Wizard
Wireless Settings
Wireless Network Name (SSID)
--- SELECT FROM LIST ---
Scan
--- SELECT FROM LIST ---
caip (WPA2, 95% signal)
JAZZTEL_GkhH (WPA2, 62% signal)
MOVISTAR_FBE0 (WPA2, 44% signal)
MOVISTAR_EA56 (WPA, 40% signal)
HP-Print-D8-Photosmart 6520 (WPA2, 38% signal)
Dulce-Hogar (WPA2, 38% signal)
caip_ext (WPA2, 32% signal)
Password
Back
Next


Arduino Configuration Wizard
Rest Api Settings
REST APIs allow you to access your sketch from the web, sending commands or exchanging configuration values. If your board is on a public network, or controlling sensitive equipment, or both, we recommend you leave the REST API password protected.
1
Back
Next


Arduino Configuration Wizard
SAVE AND RESTART
ARDUINO
Back
Save

annex 6.2 captures procés formatació targeta sd mitjançant l'*sketch* yundiskspaceexpander

primer intent havent fet totes les actualitzacions



segon intent, amb prou espai al disc

annex 6.3 s'explica com fer-ho.

Tal i com s'ha configurat el servidor -amb l'opció *uhttpd.main.index_page* - si no s'escriu un nom específic d'una plana web, el servidor mira per ordre si existeix alguna de les planes marcades per defecte i, si la troba, és la que envia per resposta. En aquest cas la plana per defecte es diu *default.htm*

El conjunt de planes que formen el portal del treball son:

- *default.htm* ⇒ Mostra les dades que es registren en el moment de fer la consulta i, també, la resta de lectures recollides durant la resta del dia.
- *arxius.php* ⇒ Mostra, organitzat per anys i mesos, el llistat dels arxius de registres de dades recollits durant l'any i mes seleccionat. Triat un dia, es mostra a *l'iframe* el contingut de l'arxiu *historic.php*
- *historic.php* ⇒ Mostra la taula amb les dades recollides durant el dia que li arriba com a paràmetre i es representa la seva evolució gràficament.
- *configuracio.php* ⇒ Mostra les dades de configuració amb les que està treballant el dispositiu i permet modificar-les.
- *enllacos.php* ⇒ Aquesta plana només és útil durant el període de desenvolupament i és un recull de les adreces interessant relacionades amb el projecte.

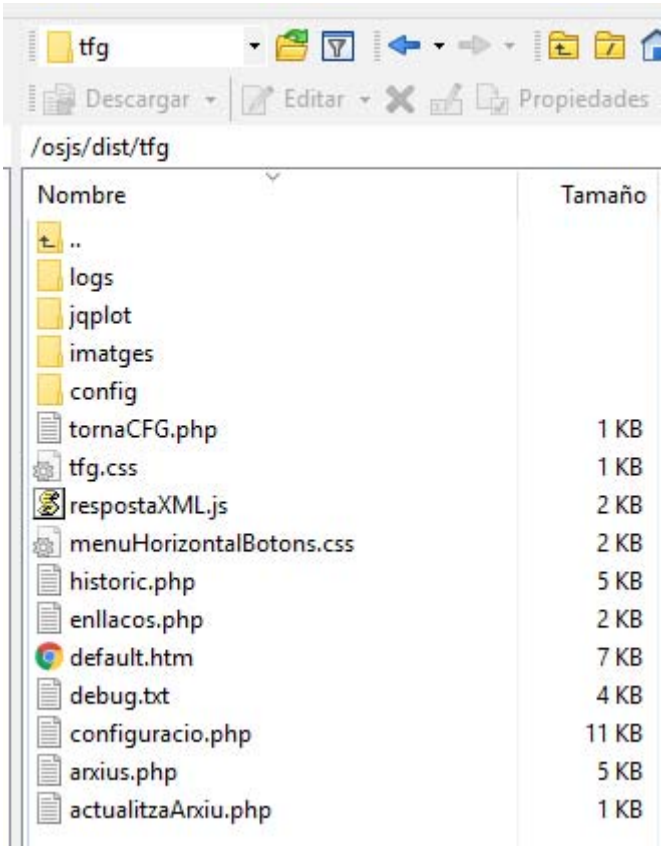
A banda d'aquestes planes, el sistema necessita d'altres arxius que es carreguen en segon terme;

- *tfg.css* ⇒ Conté els estils comuns que s'apliquen a les diferents planes
- *menuHorizontalBotons.css* ⇒ Conté els estils comuns relacionats amb els botons que permeten canviar entre les diferents planes del projecte

- *respostaXML.js* ⇒ Conté codi *javascript* comú a les diferents planes relacionat amb les peticions xmlHttpRequest de la tecnologia AJAX.
- *jqplot/*.** ⇒ Conté tots els arxius que formen el *plugin* jqPlot , un projecto gratuït de codi obert creat per Chris Leonello, que permet dibuixar gràfics a les planes web a partir d'unes sèries de dades

A més d'aquestes, també hi ha dos arxius dintre de la carpeta */tfg/config* que contenen informació sobre la configuració del sistema:

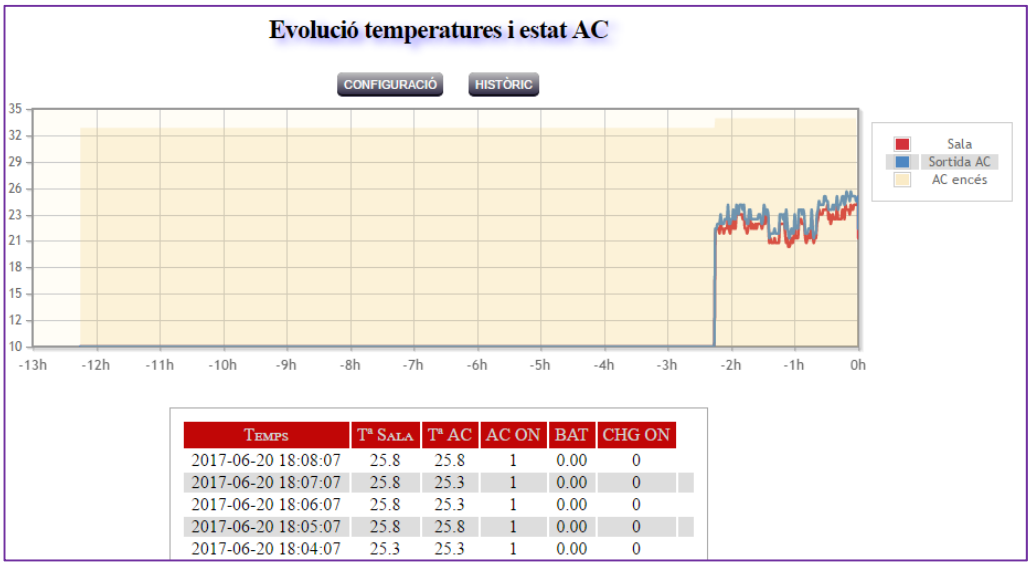
- *dadesCFG.php* ⇒ Conté les dades de configuració del sistema
- *sms_text.txt* ⇒ Conté el text que s'envia per sms quan cal fer una alerta per excés de temperatura
- *sms_dc.txt* ⇒ Conté el text que s'envia per sms quan s'ha superat el temps establert de funcionament amb bateries.
- *sms_ac.txt* ⇒ Conté el text que s'envia per sms quan es recupera el funcionament amb corrent elèctric després d'un periode de funcionament amb bateries.



Il·lustració 48 Captura de la carpeta arrel del projecte en el servidor

La plana principal, default.htm

Tal i com s'acaba de comentar, la plana principal es diu *default.htm* i té l'aspecte que es pot veure a la il·lustració següent.



Il·lustració 49 Captura pàgina web principal

En ella es poden distingir tres zones:

- Uns botons, que permeten anar a la resta de planes web del projecte
- Un gràfic, en el que es mostren l'evolució de les darreres tretze hores de les temperatures de la sala i de la sortida de l'aire condicionat, i en pinta els fons del gràfic de color beix o blanc en funció de si la comporta de l'aire condicionat està oberta o tancada.
- Una taula, que s'actualitza cada minut, que mostra de més recent a menys recent les temperatures recollides durant tot el dia.

El funcionament de la plana és el següent:

- Es carrega la plana amb el gràfic i la taula amb les dades buides.
- Mitjançant la tecnologia AJAX descrita a l'Annex 6.4, abans d'acabar de carregar la plana, es fa la lectura del l'arxiu que conté els registres de tot el dia i es posa cada filera de la taula les dades obtingudes de cada línia de l'arxiu de registre. Donat que les dades apareixen l'arxiu en ordre invers al que apareixen a la taula, el que es fa és afegir la nova filera a la taula en primera posició.
- Es redibuixa el gràfic amb les dades obtingudes
- Es programa una nova petició de dades al servidor cada minut mentre la plana estigui oberta.
- Cada minut que la plana és oberta:
    - Mitjançant la tecnologia AJAX, es fa la petició de la lectura actual al servidor, que el servidor consulta al seu torn al microcontrolador fent una consulta a l'adreça URL relativa

      `/arduino/tfg_info`
    - S'afegeix la nova lectura al capdamunt de la taula de dades.
    - Es redibuixa el gràfic.

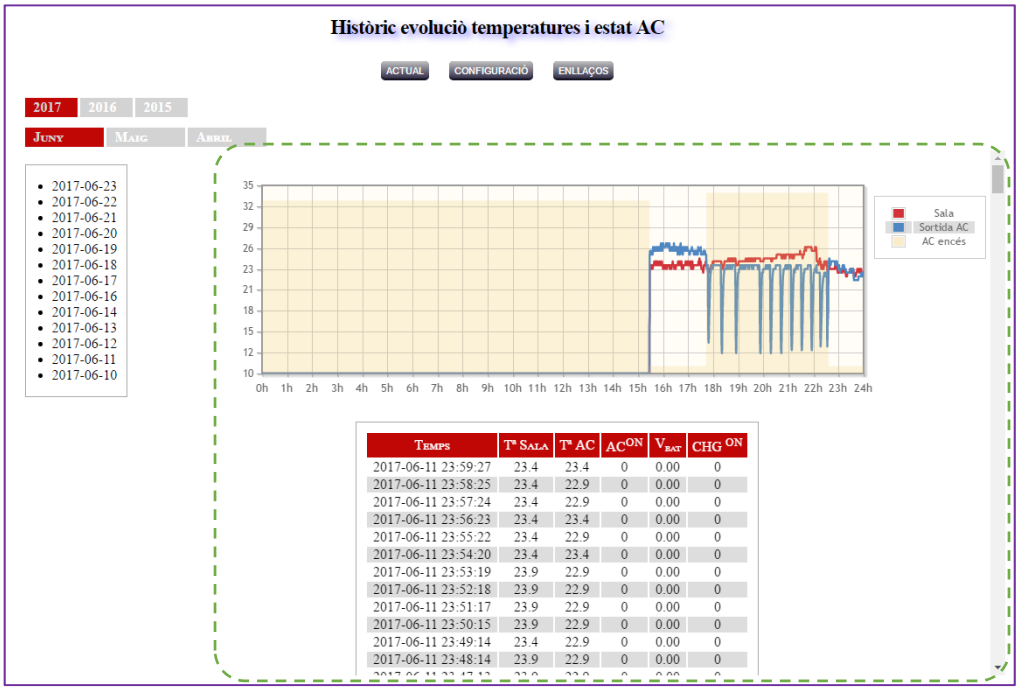
La petició de les lectures de dia es fa simplement demanant al servidor l'arxiu on es registren totes les lectures fetes durant el dia. Aquest arxiu es troba a la carpeta del servidor *logs* i el seu nom té el format *tfg_AAAA-MM-DD.txt*, on AAAA representa l'any expressat amb quatre dígits, MM

representa el mes expressat amb dos dígits i DD representa el dia del mes expressat amb dos dígits. Així doncs, la part relativa de la URL per accedir a aquests arxius té una forma semblant a:

/tfg/logs/tfg_2017-06-08.txt

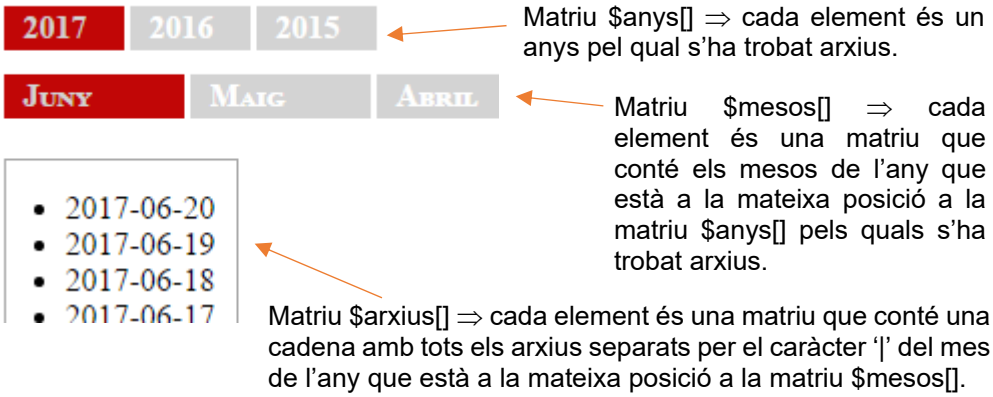
La plana de consulta de l'històric, arxius.php i historic.php

Tal i com s'ha explicat anteriorment, aquestes dues planes s'encarreguen de mostrar, organitzat per anys i mesos, el llistat dels arxius de registres de dades recollits durant l'any i mes seleccionat i, triat un dia, es mostra a l'*iframe* el contingut de l'arxiu. A la següent il·lustració podem veure la seva aparença:



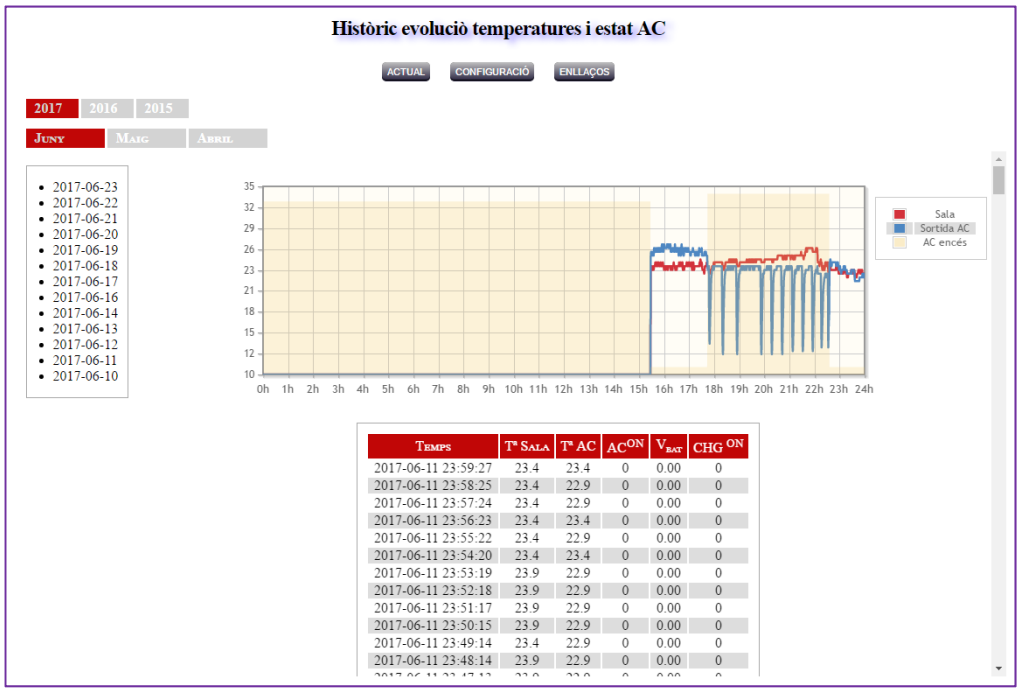
Il·lustració 50 Captura de les planes web arxius.php i historic.php

La plana *arxius.php* escaneja el directori on es desen tots els arxius de registre de dades i, analitzant el noms dels arxius –que recordem a la part central tenen la data del dia al que pertanyen les dades- omple diferents matrius que li permeten posteriorment crear els menús horitzontals amb els anys i mesos disponibles, i el menú vertical amb les dates de l'any i mes seleccionat per a les quals s'ha trobat informació.



Il·lustració 51 Detall construcció menú històric d'arxius

Fent un clic sobre la data desitjada, gràcies al javascript, a l'iframe destacat amb línies discontínues de color verd a la



il·lustració 50 es carrega la plana *historic.php* .

El funcionament de la plana *historic.php* és quasi idèntic al de la plana *default.htm* amb l'excepció que:

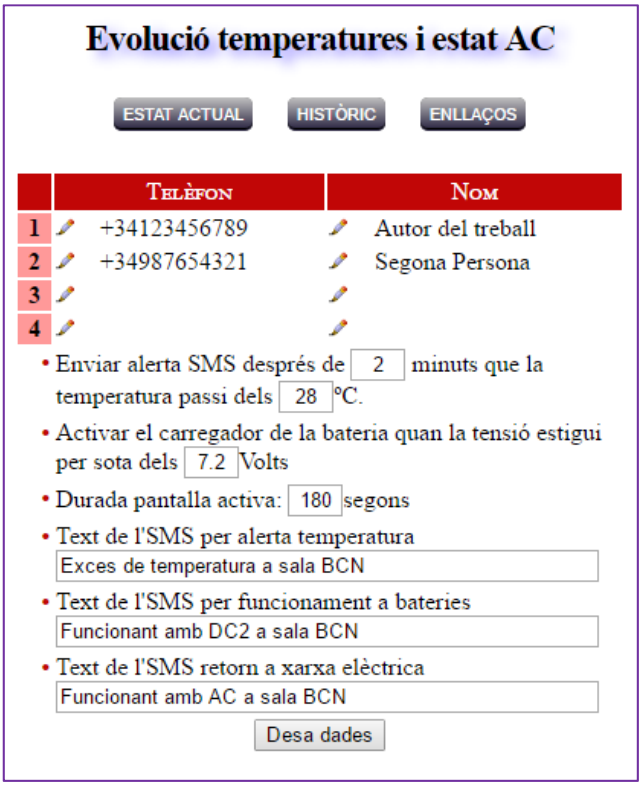
- No actualitza la informació cada seixanta segons donat que la informació que mostra no varia a cada instant.
- No representa les darreres tretze hores sinó que representa la vint-i-quatre hores del dia.

- Agafa la informació del nom de l'arxiu que ha d'obrir del camp data que li arriba com a paràmetre en el moment de cridar la plana.

La plana de configuració del sistema, configuracio.php

Aquesta plana permet visualitzar i modificar la informació de configuració del dispositiu sense necessitat de modificar el seu codi font. Dels canvis en la configuració:

- El servidor web s'encarrega de desar-los en els arxius corresponents en la carpeta */tfg/config*
- Mitjançant una crida REST, es dóna l'ordre al micro controlador per tal que llegeixi l'arxiu de configuració i la posi en memòria.

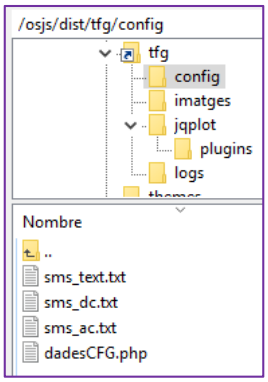


Il·lustració 52 Captura plana web configuracio.php

Tal i com es pot apreciar a la il·lustració anterior aquesta plana permet:

- EDITAR UNA PETITA AGENDA de fins a 4 telèfons als qui es notificaran per SMS les incidències que detecti el sistema.
- DETERMINAR EL LLINDAR DE LA TEMPERATURA per sobre de qual s'enviarà un SMS d'alerta als integrants de la llista

- Establir durant quan de TEMPS HA D'ESTAR L'ALARMA ACTIVADA per tal que es consideri que cal notificar-ho per SMS
- DETERMINAR EL LLINDAR DE LA TENSIÓ per sota del qual s'activarà el carregador durant 5 hores.
- DETERMINAR ELS MINUTS que estarà la PANTALLA ACTIVA abans que entri en funcionament el protector de pantalla per tal d'allargar la seva vida útil.
- PERSONALITZAR EL TEXT DEL MISSATGE SMS que es rebrà quan calgui avisar D'EXCÉS DE TEMPERATURA.
- PERSONALITZAR EL TEXT DEL MISSATGE SMS que es rebrà quan calgui avisar de FUNCIONAMENT AMB BATERIES.
- PERSONALITZAR EL TEXT DEL MISSATGE SMS que es rebrà quan calgui avisar de RETORN A FUNCIONAMENT AMB CORRENT ELÈCTRIC.



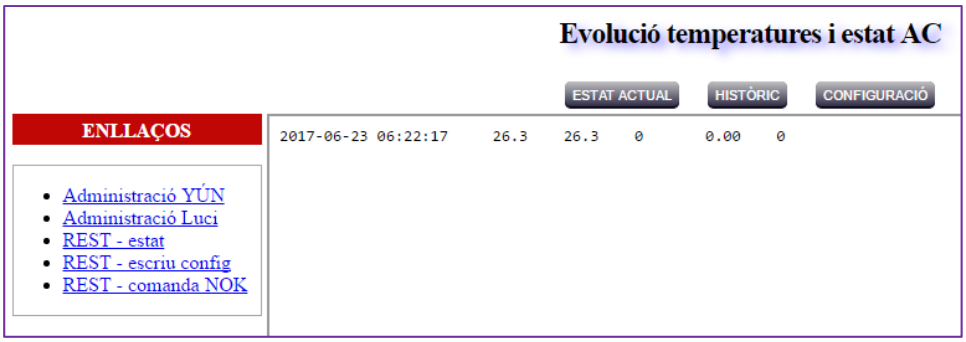
Il·lustració 53 Captura carpeta amb arxius de configuració

Tal i com es pot veure en la il·lustració anterior, tota aquesta informació es desa en quatre arxius diferents: un arxiu per a cada missatge SMS i un altre on es desen la resta de dades.

Aquest arxiu amb dades és un arxiu *php* en el que hi ha una única línia de codi de tipus comentari amb totes les dades separades entre dos caràcters '|' consecutius. En ser una línia de codi, si s'intenta descarregar directament l'arxiu des de la xarxa el que apareix és una plana en blanc. D'aquesta manera es protegeix mínimament l'agenda telefònica.

La plana d'enllaços d'ajuda al desenvolupament, enllacos.php

Ja s'ha comentat que aquesta plana s'ha creat a efectes de facilitar les tasques de verificacions durant el desenvolupament del projecte i només és tracta d'un recull de les adreces interessants relacionades amb el projecte.



Il·lustració 54 Aspecte de la plana enllacos.php

La plana conté un enllaç per a cadascun dels portals d'administració i un enllaç per a cadascuna de les planes per les que s'ha configurat el microcontrolador per tal que doni resposta. A la figura anterior es veu el resultat d'haver fet una consulta al micro controlador per tal que informi de l'estat dels sensors.

2.4.10 El *sketch* d'Arduino

El codi desenvolupat pel projecte ocupa la totalitat de l'espai d'emmagatzematge del programa la memòria. Exactament ocupa 28594 bytes – un 99%- dels 28672 disponibles.

Pel que fa a les variables globals, aquestes ocupen 1374 bytes – un 53%- de la memòria dinàmica. Aquestes bàsicament s'organitzen en quatre estructures de dades segons la informació que contenen:

- Una estructura que conté les variables amb la informació de l'estat del sistema.

- Una estructura que conté les variables amb la configuració dels sistema
- Una estructura que conté les variables que contenen informació de control sobre accions que ha de fer el sistema.
- Una estructura que conté els número de telèfon de l'agenda

```
struct dadesEstat {
   unsigned int iTempS;
   unsigned int iTempAC;
   unsigned int iV2;

   boolean bAireCondicionat_ON=false;
   boolean bAmbDC=false;
   boolean bAlertaT=false;
   boolean bAlertaDC=false;
   boolean bMostrarError=false;
   boolean bGSM_ON=false;
   boolean bCarregant=false;
};
struct agenda_str {
   struct str_telf telf[4];
};
```

```
struct dadesAccions {
  unsigned long ulProperMostreig;
  unsigned long ulProperLog;
  unsigned long ulPantallaOFF;  //##
  unsigned long ulIniciAlertaT;
  unsigned long ulIniciAlertaDC;
  unsigned long ulFiCarrega;
};

struct dadesConfig {
   unsigned char  iPantallaOFF=180;
   unsigned char  iTAlerta=24;
   unsigned char  iLatenciaSMS=10;
   unsigned char  iCarregadorON=60;

};
```

Il·lustració 55 Estructures variables globals de l'sketch

S'utilitzen set llibreries diferents:

- Bridge.h, BridgeServer.h i BridgeClient.h ⇒ que conten les funcions necessàries per a la interacció en el micro controlador i el microprocessador
- FileIO.h ⇒ que conté les funcions per accedir al sistema d'arxius de la targeta SD
- genieArduino.h ⇒ que conté les funcions que gestionen la interacció amb la pantalla
- SoftwareSerial.h ⇒ que conté les funcions per gestionen la comunicació sèrie amb el mòdul GSM
- AltSoftSerial.h ⇒ que conté les funcions per gestionen la comunicació sèrie amb la pantalla

Tot el codi –es pot consultar a l'

annex 6.5-està dividit en múltiples funcions que faciliten la comprensió i, sobretot, compacten al màxim el codi per tal de minimitzar la memòria que ocupa. Donat que el codi ja està força explicat en sí mateix, en aquest apartat només se'n destacaran els aspectes principals.

El bucle principal del programa del micro controlador fa només 4 coses:

- GESTIONAR ELS ESDEVENIMENTS provinents de la pantalla tàctil, cas d'haver algun esdeveniment a la cua.
- MOSTREJAR ELS VALORS DELS SENSORS.
    - Es mostregen els sensors.
    - S'actualitza la informació de la pantalla.
    - Es comprova si cal disparar alguna alerta o si s'ha de posar les bateries a carregar
    - Es desa la informació de l'estat a l'arxiu de registre.
- ATENDRE PETICIONS REST del servidor web del micro controlador.
- ACTIVAR EL PROTECTOR de la pantalla si s'ha superat el temps d'inactivitat establert.

Les pulsacions sobre la pantalla arriben a l'Arduino en forma d'esdeveniments a gestionar que s'ordenen en una cua. Des del gestor d'aquesta cua s'inicia l'execució de la funció oportuna, si s'escau, en el microcontrolador.

Donat que la temperatura no és una magnitud que variï a una gran velocitat, l'autor ha decidit prendre una lectura dels sensors cada cinc segons. Aquesta lectura només serveix per a actualitzar la visualització a la pantalla i evidenciar d'aquesta forma que el sistema està funcionant. Tot i fer lectures cada cinc segons, a l'arxiu que conté el registre de les mesures només es desa una lectura a cada minut. Aquestes lectures son les que serveixen per a dibuixar el gràfics amb l'evolució des de l'entorn web.

Quan es detecta que la temperatura ha sobrepassat el llindar prefixat, el que es fa és activar la variable *estat.bAlertaT* i posar una marca de temps

a la variable *estat.ulIniciAlertaT* . Si passat el temps estipulat en la configuració aquesta variable segueix activa, s'envia l'SMS als números existents a l'agenda. Si durant aquest període l'alarma desapareix, no s'envia cap missatge.

De manera equivalent funciona el mecanisme de funcionament amb bateries. Si els sistema treballa amb bateries s'activa la variable *estat.bAlertaDC* i es posa una marca de temps a la variable *estat.ulIniciAlertaDC* . Si passat el temps estipulat en la configuració aquesta variable segueix activa, s'envia l'SMS als números existents a l'agenda. Si durant aquest període l'alarma desapareix, no s'envia cap missatge. Si desapareix però s'havia enviat un missatge, aleshores s'envia un altre missatge informant que s'ha recuperat el funcionament amb el subministrament elèctric.

El tercer estat del que es té cura és el de la tensió a les bateries i de si cal posar en marxa el carregador. Això es fa activant la variable *estat.bCarregant* i posant una marca de temps a la variable *estat.ulFiCarrega* que indica al sistema quan s'ha d'aturar el carregador.

Les peticions REST s'han limitat a retornar l'estat dels sensors en el darrer mostreig fet i a llegir el fitxer de configuració i carregar-lo a la memòria en el moment de rebre aquesta petició.

La protecció de pantalla s'ha posat per dos motius. D'una banda el fabricant recomana no mostrar durant llargs períodes de temps la mateixa informació a la pantalla. D'altra banda, el LED que retro il·lumina la pantalla també té una durada determinada i apagant la pantalla es pot allargar la seva vida útil.

Quan es tanca la pantalla el que es fa és anar a una que té un botó ocult que ocupa tota la superfície i després posar tancar el LED. Quan es toca la pantalla, internament el que s'està fent és pitjar el botó que la reactiva.

3. Valoració del projecte

3.1 El pressupost

El cost del producte ha resultat elevat sobretot degut al cost de la pantalla utilitzada –més d'un terç del total-. Si en comptes d'aquest acabat se'n volgués un de més senzill, es podria substituir aquest display per un de més senzill.

En aquest projecte s'ha preferit disposar d'aquesta pantalla per tal de poder explotar les seves capacitats en futures ampliacions.

Preu	Unitats	Element
66,49 €	1	Arduino YUN
136,75 €	1	LCD Display 4.3" 480x272 C/Touch Arduino
39,99 €	1	Mòdul SIMCOM SIM900 GSM GPRS de Kuman
5,46 €	1	Font d'alimentació Mean Well RS-25-5
23,79 €	2	Bateries lipo 7,4v 2000mAh + carregador
6,60 €	2	Mòdul de relé de 2 canals amb 5V
5,02 €	1	Mòdul step down DC-DC (Velleman VMA404)
2,88 €	2	Sensor temperatura LM35
2,07 €	1	Mini magnetic reed module (Velleman VMA308)
289,05 €		***Subtotal components***
12,96 €	1	Caixa plastibox PP-39N 200x110x73)
2,60 €	1	Paca soldadura
1,16 €	8	Separadors hexagonals M-H-N 8mm
1,16 €	8	Separadors hexagonals M-H-N 5mm
1,31 €	1	Tira 40 pins mascle PCB
1,44 €	1	Tira 40 pins mascle acodada PCB
1,61 €	1	Tira 40 pins femella PCB
1,09 €	1	Base alimentacio xasís 1,9x5,5
3,44 €	1	Roseta 1x UTP CAT5
5,08 €	1	Base hembra RJ45 CAT5
1,63 €	0,5	Placa Pet incolor 3mm DINA4
1,95 €	1	Cable antena SMA M - SMA H 1m (Valueline VGSP02010B10)
3,64 €	1	Commutador bipolar basculante
0,77 €	2	Connector mascle DC power Plug jack (2,1*5,5mm)
0,38 €	1	Connector famella DC power Plug jack (2,1*5,5mm)
0,32 €	2	Fusible de fusió ràpida de 2.5A
12,22 €	2	Porta fusible de pinça per circuit integrat
52,75 €		***Subtotal acabat prototip comercial***
341,80 €		TOTAL

Il·lustració 56 El pressupost

3.2 VIABILITAT DEL PROJECTE

Tot i que el preu és una mica elevat, no s'allunya massa d'altres productes que s'ha trobat per la xarxa amb prestacions semblants, amb l'avantatge que aquest producte és personalitzable. Donat que aquest projecte està dissenyat com a possible solució per a consum intern de l'organització en les seves diferents seus, el cost no és un requeriment crític del projecte.

Si es volgués anar més enllà i comercialitzat el producte, caldria fer més investigacions ja que la capacitat de programació de l'Arduino Yún no permet més línies de codi i qualsevol canvi és crític. Caldria mirar quines possibilitats ofereixen altres plaques i quan canvis caldria fer.

3.3 LES CONCLUSIONS

Vaig començar aquest projecte amb molta il·lusió ja que, tot i que no havia tingut cap contacte directe, havia sentit parlar del món Arduino i tenia ganes de poder experimentar amb ell. Inicialment semblava que oferia múltiples possibilitats i que era relativament fàcil anar interconnectant diferents dispositius.

Durant la fase de recerca vaig comprovar que efectivament hi ha un munt de possibilitats i coses a fer, però em vaig trobar que en alguns camps la cosa no era tant fàcil. Vaig creure, per exemple, que disposar d'un sistema d'alimentació ininterrompuda seria només qüestió de triar entre un munt de possibilitats en funció del criteri que cregués més oportú. Per sorpresa em vaig trobar que pràcticament no hi havia cap mòdul que fes aquesta funció i que per tant calia implementar-ho personalment. Cal dir, però que amb la resta de mòduls va passar el contrari. Hi havia molt on escollir i només era qüestió de triar el que millor s'adaptés al projecte.

Un altre aspecte que m'ha sorprès ha estat la dificultat en organitzar tots els cables i elements que formen el projecte i mirar d'encabir-los en una caixa de forma estable i desmuntable. Vull dir amb això que una cosa és anar connectant cables quan el projecte va creixent i una altra de molt diferent és tenir-los organitzats i saber on va cada cosa sense haver de resseguir tot l'esquema de nou.

Per últim, una qüestió fàcil per a mi -donada la meva experiència en el camp- i que s'ha complicat enormement en el projecte ha estat la programació. No he tingut cap problema mentre he anat provant les coses per separat. El problema has estat en el moment d'integrar-ho tot. L'Arduino no disposa de massa espai per tal de poder programar un sistema i fer-lo versàtil. El fet de poder personalitzar la configuració del sistema ocupa força línies de codi i limita les possibilitats de coses a fer.

He tingut problemes de comportaments erràtics del sistema que només s'han solucionat en alliberar memòria dinàmica dedicada a variables globals i deixant així més espai per a variables locals.

Malgrat tot, el producte tal i com està, compleix els objectius establerts a l'inici del projecte i a més permet una anàlisi visual de la informació recollida. El funcionament del doble canal d'accés al sistema, tant mitjançant la pantalla tàctil com per l'entorn web, em deixa especialment satisfet.

El fet d'haver superat tots els obstacles que han anat apareixent fins a aconseguir un producte que compleix amb el que m'havia plantejat em deixa esgotat però satisfet amb la feina feta.

4. Glossari

Adreça IP ⇒ D'acord amb el protocol d'Internet, una adreça IP és un nombre que identifica inequívocament un dispositiu lògic connectat a la xarxa. Dins d'una mateixa xarxa, cada adreça IP que s'utilitzi ha de ser única.

Arduino ⇒ és una placa de circuit imprès simple basada en el microcontrolador de codi obert provinent de la plataforma de codi obert amb l'objectiu de fer més simple i accessible el disseny de circuits electrònics amb microcontroladors.

AJAX (*Asynchronous JavaScript And XML*) ⇒ és una tècnica de desenvolupament web que ens permet tenir comunicacions asíncrones amb el servidor de manera que aconseguim recarregar porcions de la pàgina activa, sense necessitat de recarregar-la sencera.

PHP ⇒ llenguatge informàtic que s'utilitza principalment per a generar pàgines web de forma dinàmica. S'executa al cantó del servidor, per aquest motiu al navegador web ja l'hi arriba la pàgina en format HTML, no podent visualitzar-ne el codi php.

REST (*REpresentational State Transfer*) ⇒ és un protocol d'intercanvi i manipulació de dades en els serveis basats en hipermèdia, com ara el web.

Sketch ⇒ nom que reben cadascun dels conjunts de línies de codi que formen un programa elaborat per funcionar amb un micro processador Arduino**.**

URL (*Uniform Resource Locator*) ⇒ cadena de caràcters – o adreça d'Internet- que informa al navegador de la màquina on és el recurs a què fa referència, el protocol que s'ha d'utilitzar per

obtenir aquest recurs i de la manera com el servidor web trobarà quin és el recurs.

Wiring ⇒ plataforma de prototipatge de programari obert compost per un llenguatge de programació, una interfície de desenvolupament integrada i una placa base amb un microcontrolador. El seu desenvolupament va començar el 2003 per part de Hernando Barragán**.**

5. Bibliografía

4D Systems. (març de 2017). Obtenido de Arduino Adaptor Shield II: http://www.4dsystems.com.au/product/4D_Arduino_Adaptor_Shield_II/

4D Systems. (març de 2017). Obtenido de gen4-uLCD-43D Series: http://www.4dsystems.com.au/product/gen4_uLCD_43D/

Arduino. (abril de 2017). *How to expand the Yún disk space*. Obtenido de https://www.arduino.cc/en/Tutorial/ExpandingYunDiskSpace

Instalación y configuración de PHP en el Arduino Yún. (04 de 2017). Obtenido de Panama hitek: http://panamahitek.com/instalacion-de-php-arduino-yun/

jqPlot. (abril de 2017). Obtenido de http://www.jqplot.com/

LUIS LLAMAS. (abril de 2017). Obtenido de Ingeniería, informática y diseño: https://www.luisllamas.es/calculadora-divisor-de-tension/

LUIS LLAMAS. (abril de 2017). Obtenido de Ingeniería, informática y diseño: https://www.luisllamas.es/referencia-lenguaje-arduino/

OKAWA Electric Design. (abril de 2017). *RC Low-pass Filter Design Tool*. Obtenido de http://sim.okawa-denshi.jp/en/CRtool.php

OpenWrt. (abril de 2017). *OpenWrt Wireless Freedom*. Obtenido de https://openwrt.org/

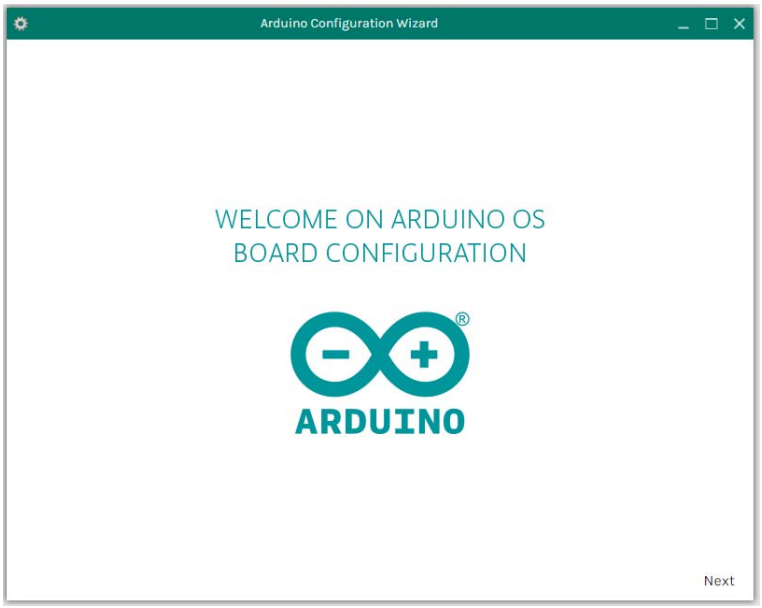
Putty.org. (abril de 2017). Obtenido de http://www.putty.org/

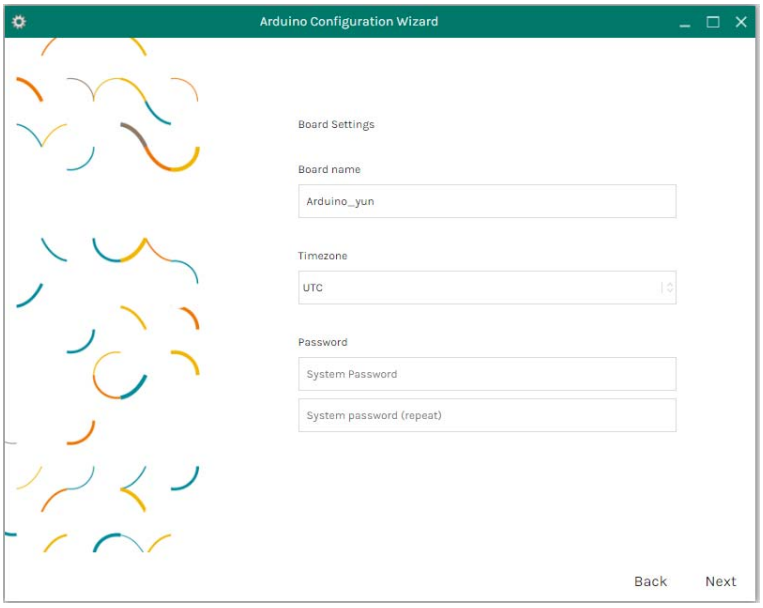
WinSCP. (abril de 2017). Obtenido de https://winscp.net/eng/docs/lang:ca

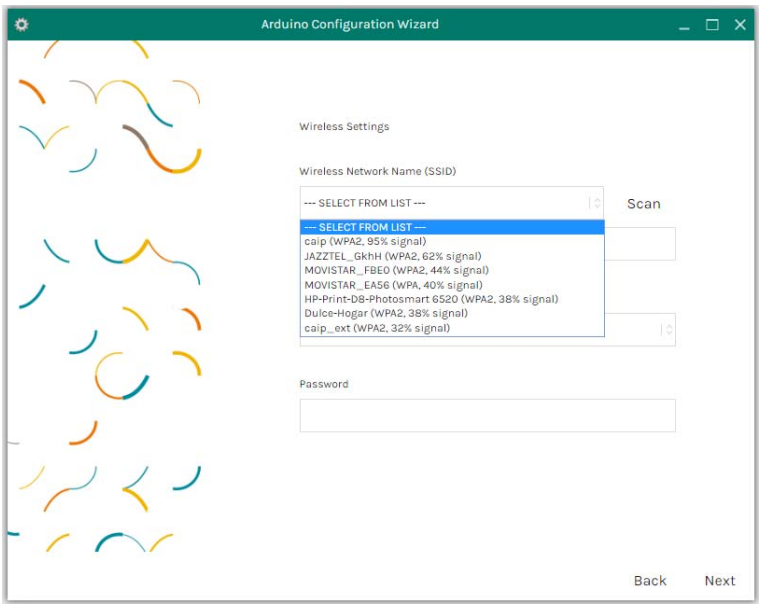
www.lucadentella.it. (abril de 2017). Obtenido de http://www.lucadentella.it/en/2013/11/12/yun-utilizziamo-una-scheda-sd/

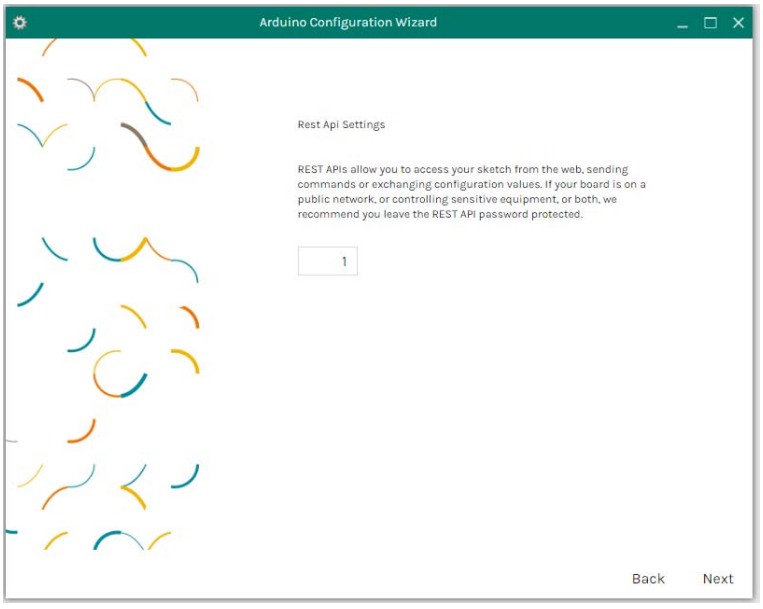
Annexes

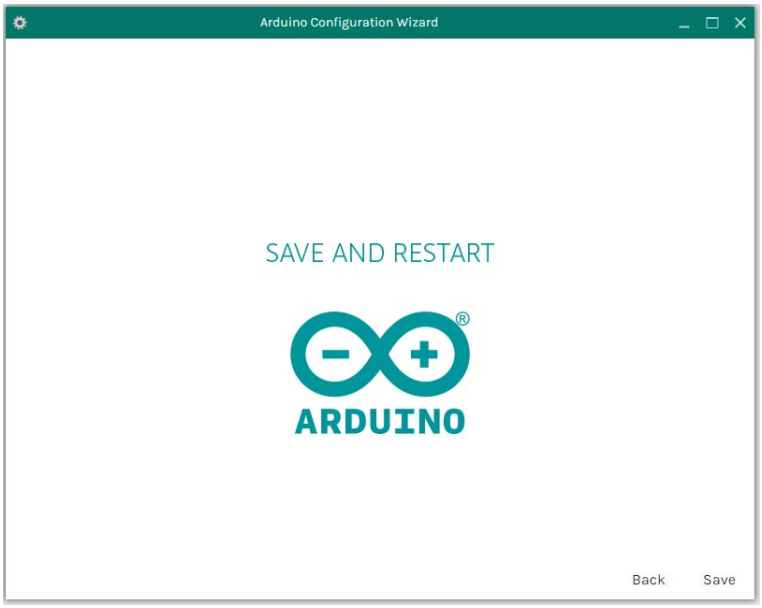
Annex 6.1 Captures del procés de configuració inicial de l'Arduino Yún





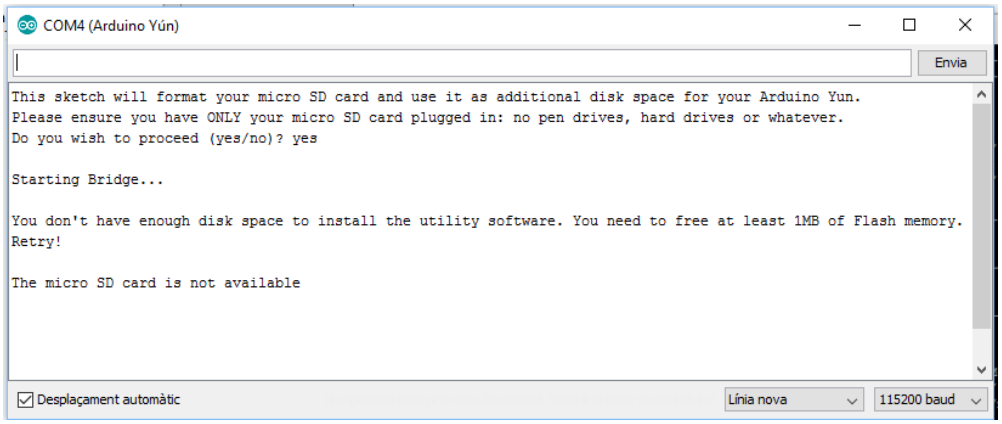

Arduino Configuration Wizard
Wireless Settings
Wireless Network Name (SSID)
--- SELECT FROM LIST ---
Scan
--- SELECT FROM LIST ---
caip (WPA2, 95% signal)
JAZZTEL_GkhH (WPA2, 62% signal)
MOVISTAR_FBE0 (WPA2, 44% signal)
MOVISTAR_EA56 (WPA, 40% signal)
HP-Print-D8-Photosmart 6520 (WPA2, 38% signal)
Dulce-Hogar (WPA2, 38% signal)
caip_ext (WPA2, 32% signal)
Password
Back
Next


Arduino Configuration Wizard
Rest Api Settings
REST APIs allow you to access your sketch from the web, sending commands or exchanging configuration values. If your board is on a public network, or controlling sensitive equipment, or both, we recommend you leave the REST API password protected.
1
Back
Next

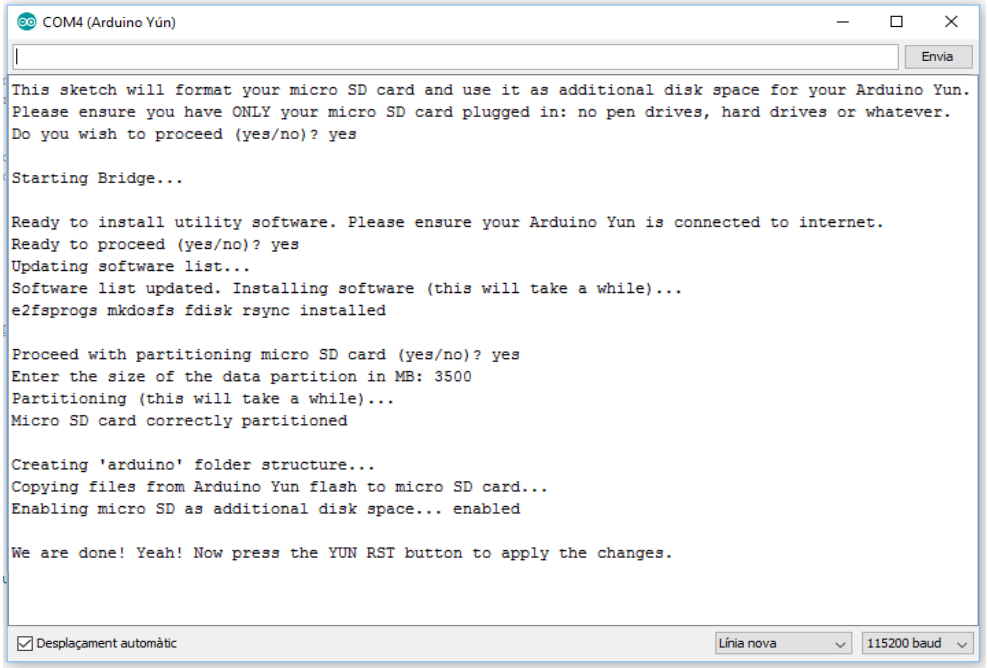

Arduino Configuration Wizard
SAVE AND RESTART
ARDUINO
Back
Save

Annex 6.2 Captures procés formatació targeta SD mitjançant l'*sketch* YunDiskSpaceExpander

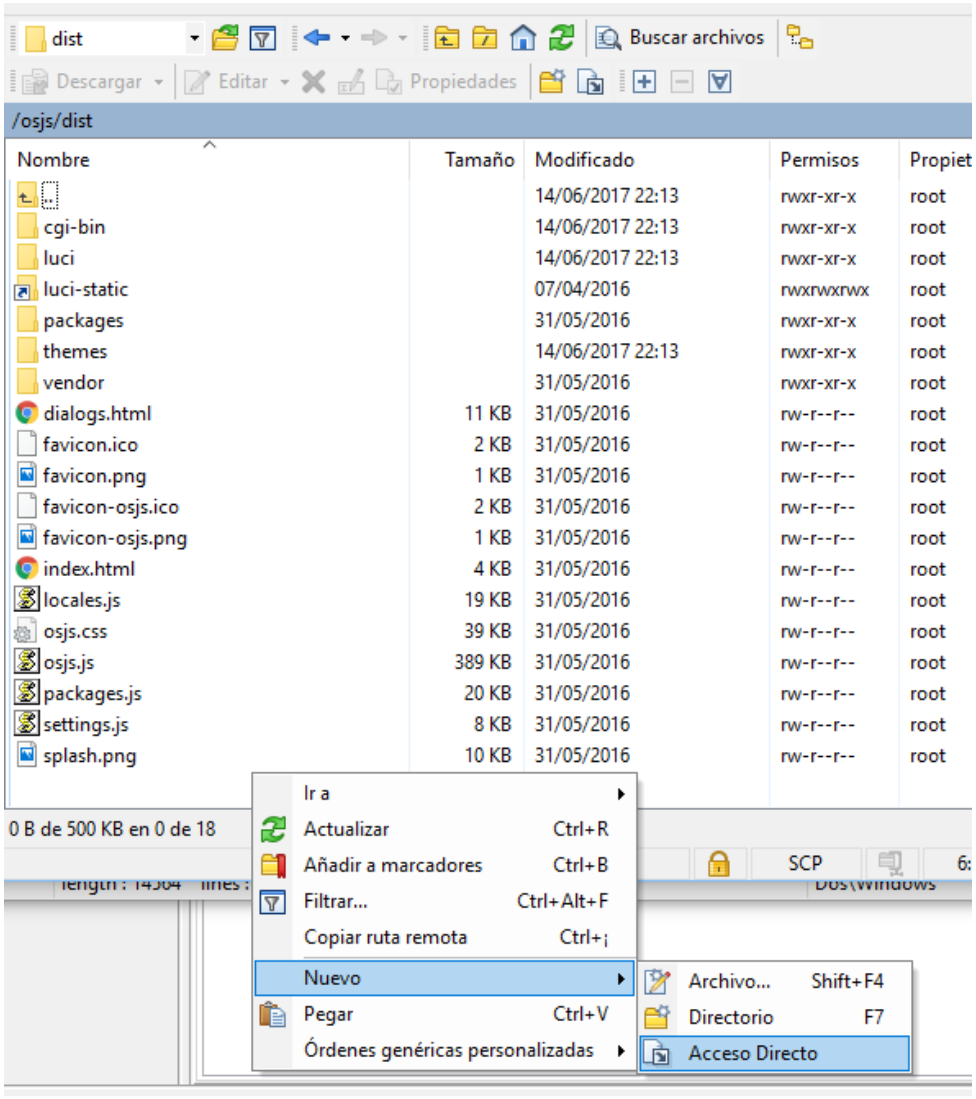
Primer intent havent fet totes les actualitzacions

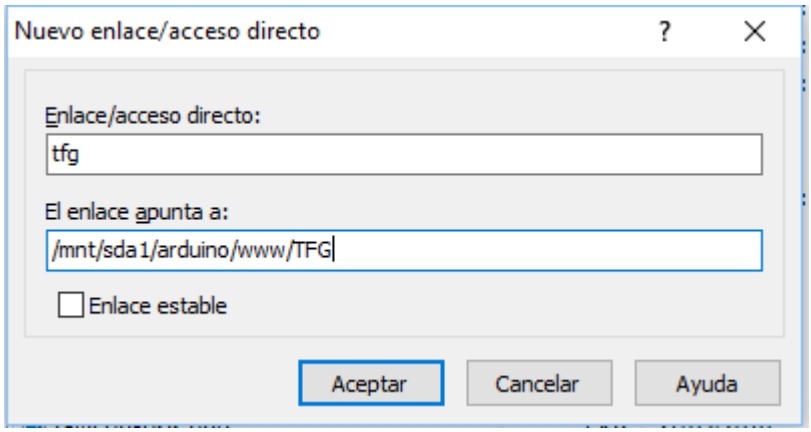


Segon intent, amb prou espai al disc



Annex 6.3 Captures creació enllaç directe al portal en el directori arrel del servidor web





Annex 6.4 Funcionament emprat de la tecnologia AJAX

Recordem que AJAX és una tècnica de desenvolupament web que ens permet tenir comunicacions asíncrones amb el servidor de manera que aconseguim recarregar porcions de la pàgina activa, sense necessitat de recarregar-la sencera.

Això es fa mitjançant un objecte que els navegadors moderns ja incorporen i que s'anomena *XMLHttpRequest*.

Aquest objecte permet demanar informació a un servidor i processar la seva resposta. En aquest projecte no ha calgut fer una interpretació XML del resultat retornat i només ha calgut interpretar els resultats obtinguts en mode text pla.

Per utilitzar-lo només cal definir un objecte d'aquesta classe:

```
xmlhttp=new XMLHttpRequest();
```

Aquest objecte té un mètode *open()* que permet demanar el recurs desitjat al servidor. Aquesta crida es pot fer de forma síncrona o asíncrona. Per tal que el funcionament sigui AJAX, sempre hem de triar el mètode asíncron.

xmlhttp.open(*method, url, async*)

```
xmlhttp.open("GET", "xmlhttp_info.txt", true);
```

Un cop definit a on i com ens volem connectar, aleshores cal iniciar la petició fet ús del mètode *send()*.

Abans de cridar al mètode *send()* cal especificar què farem en el moment que haguem rebut la resposta a la crida feta al servidor. Això es fa mitjançant definint la resposta a l'esdeveniment *onreadystatechange.*

Si la propietat *readyState* té un valor de 4 vol dir que ja ha arribat tota la resposta del servidor. Aleshores cal llegir la propietat *status* per saber si

tot ha arribat correctament –la propietat tindrà un valor de 200- o hi ha hagut algun problema. En el nostre cas el resultat l'emmagatzemem a una variable global anomenada *respostaXML*. Si hi ha hagut algun error, a l'inici d'aquesta variable posarem els caràcters "#~#" juntament amb el codi de l'error.

```
xmlhttp.onreadystatechange=function()  {
    if (xmlhttp.readyState==4) {
        respostaXMLEnEspera=0;
        switch (xmlhttp.status) {
            case 200: respostaXML=xmlhttp.responseText; break;
            case 400: respostaXML='#~# Error 400 #~#<br>'+xmlhttp.responseText;  break;
            default: respostaXML='#~# codi' +  xmlhttp.status +' #~#<br>'+xmlhttp.responseText;
        }
    }
}
```

Amb tot això l'única cosa que ens manca és crear un temporitzador que a intervals regulars comprovi si ja hm rebut les dades i, en cas d'haver-ho fet, gestioni la informació rebuda de la manera més adient en cada cas.

```
var readyStateCheckInterval = setInterval(function() {
    if (respostaXMLEnEspera == 0) {
        clearInterval(readyStateCheckInterval);
        if (respostaXML.substr(0,3)!="#~#") {
            //## No hi ha error &raar; a respostaXML tinc el text retornat
            mostraDades(respostaXML);
        } else {
            //## Hi ha error a la resposta
            console.log('resposta NOK');
            //demanaDades();
        }
    }
},1000);
```

En aquest cas la comprovació es fa un cop cada segon –o mil mil·lisegons-

Annex 6.5 Sketch o codi font del programa

```
#include <Bridge.h>
#include <BridgeServer.h>
#include <BridgeClient.h>

#include <FileIO.h>

#include <genieArduino.h>
#include <AltSoftSerial.h>
#include <SoftwareSerial.h>

//## Pins per comunicació amb pantalla
#define Rx_P 13
#define Tx_P 5
#define RESET_P 4

//## Pins per comunicació amb mòdul oSIM900
#define Rx_S 7
#define Tx_S 8
#define SIM900_OnOff 9

//## Pins dels sensors i del actuadors
#define SENSOR_TENSIO  A0
#define SENSOR_T1  A1
#define SENSOR_T2  A2
#define SENSOR_REED  12
#define SENSOR_AC  6
#define RELE_4  11
#define RELES_1_2 10
#define V2 7.4     // Tensió nominal de les bateries

#define RUTA_WWW "/mnt/sda1/arduino/www/TFG/"

struct dadesEstat {
   unsigned int iTempS;
   unsigned int iTempAC;
   unsigned int iV2;

   boolean bAireCondicionat_ON=false;
   boolean bAmbDC=false;
   boolean bAlertaT=false;
   boolean bAlertaDC=false;
   boolean bMostrarError=false;
   boolean bGSM_ON=false;
   boolean bCarregant=false;
};

struct dadesAccions {
  unsigned long ulProperMostreig;
  unsigned long ulProperLog;
  unsigned long ulPantallaOFF;  //## Instant detecció alerta temperatura
  unsigned long ulIniciAlertaT;
  unsigned long ulIniciAlertaDC;
  unsigned long ulFiCarrega;
};
```

```
55
56  struct dadesConfig {
57     unsigned char  iPantallaOFF=180;
58     unsigned char  iTAlerta=24;
59     unsigned char  iLatenciaSMS=10;
60     unsigned char  iCarregadorON=60;   //## Cal dividir per 10 per tenir el float
61
62  };
63
64  struct str_telf {
65    char numero[12]="";
66    char nom[40]="";
67  };
68
69  struct agenda_str {
70     struct str_telf telf[4];
71  };
72
73  // OBJECTES
74  BridgeServer oHTTP;
75  Genie oLCD;
76  AltSoftSerial oPantalla(Rx_P, Tx_P);
77  SoftwareSerial oSIM900(Rx_S, Tx_S); // Configura el puerto serial para el oSIM900
78
79  // VARIABLES
80  //unsigned long ulMilisInici;
81  //String sDataHoraInici;
82  String sTmp="";
83
84
85  struct dadesConfig config;
86  struct dadesEstat estat;
87  struct dadesAccions accions;
88  struct agenda_str agenda;
89
90
91  void setup() {
92    FileSystem.begin();
93    Bridge.begin();
94
95    // Velocitats de comunicacio
96    oPantalla.begin(19200);     //## Per la comunicació a la pantalla s'obre una connexió serial
97    oLCD.Begin(oPantalla);      //##
98    oSIM900.begin(19200);       //## Configura velocidad serial para el oSIM900
99
100   //## PINS ENTRADES
101   pinMode(SENSOR_TENSIO,INPUT);
102   pinMode(SENSOR_T1,INPUT);
103   pinMode(SENSOR_T2,INPUT);
104   pinMode(SENSOR_REED,INPUT);
105   pinMode(SENSOR_AC,INPUT);
106
107   //## PINS SORTIDES
108   pinMode(RELE_4,OUTPUT);
109   pinMode(RELES_1_2,OUTPUT);
110   pinMode(SIM900_OnOff,OUTPUT);
111
112   //## Desactivació relés que  aïllen les bateries durant la càrrega
113   digitalWrite(RELES_1_2, HIGH);
114   digitalWrite(RELE_4, HIGH);
115
116   //## El fabricant recomana fer un reset a la pantalla abans de començar
117   resetPantalla();
118
119   oLCD.WriteObject(GENIE_OBJ_TANK, 0x00, 40);     //## Actualitza estat tank
120   escriuLCD(9, F("Arduino iniciat\nPantalla iniciada\nIniciant GSM")); //## Actualitza missatge progrés
121   escriuLCD(11, F("*\n*\n*"));                                 //## Actualitza missatge progrés
122
123   carregaCFG(); //## Omplo estructura config
124
125   encenSIM900();
126   oLCD.WriteObject(GENIE_OBJ_TANK, 0x00, 80);     //## Actualitza estat tank
127   escriuLCD(9, F("Arduino iniciat\nPantalla iniciada\nGSM iniciat"));   //## Actualitza missatge progrés
128
129   //## S'inicia l'escolta de peticions http entrants,
130   //## però únicament provinents de la xarxa interna
131   oHTTP.listenOnLocalhost();
132   oHTTP.begin();
133
```

```
  oLCD.WriteObject(GENIE_OBJ_TANK, 0x00, 100);     //## Actualitza estat tank
  escriuLCD(9, F("Arduino iniciat\nPantalla iniciada\nGSM iniciat\nServidor web iniciat"));  //## Actualitza missatg
  escriuLCD(11, F("*\n*\n*\n*"));                       //## Actualitza missatge progrés

  //## Inicia algunes variables globals
  //sDataHoraInici=tornaDataHora('T');        //## Guarda la data i l'hora en la que s'ha iniciat el sistema
 // ulMilisInici=millis();             //## Guarda els mil·lisegons que porta el sistema encés
  accions.ulPantallaOFF = tornaMarcaTemps(config.iPantallaOFF);   //## Temporitzador per tancar la pantalla

  delay(5000);

  oLCD.WriteObject(GENIE_OBJ_FORM , 0x01, 1); //## Activa formulari menú

  accions.ulProperMostreig=millis(); //## Instant en que es durà a terme el
                  //##  el proper mostreig del sensors
  accions.ulProperLog= accions.ulProperMostreig; //## Instant en que es desarà un nou
                 //##  LOG a l'arxiu

  estat.bAmbDC=(digitalRead(SENSOR_AC)== HIGH  ? false : true);
  oLCD.WriteObject(GENIE_OBJ_LED, (estat.bAmbDC ? 0x01 : 0x00), 1);   //## AC ON
  oLCD.AttachEventHandler(myGenieEventHandler);
}




//  ////////////////////////////////////////////
//                  BUCLE PRINCIPAL
//  ////////////////////////////////////////////

void loop() {
  oLCD.DoEvents();              //## Gestiona cua d'events provinents
                  //##  de la pantalla
  if (millis() >= accions.ulProperMostreig)
    mostrejaSensors();

    servidorWeb();       //## Si li demanen, el servidor retorna
             //##  la darrera lectura dels sensors

  if (millis() > accions.ulPantallaOFF )
  salvaPantalla();          //## Per estalviar energia, passat un
                     //##  temps es tanca la pantalla
}




//  ////////////////////////////////////////////
//                     CONFIG
//  ////////////////////////////////////////////
void carregaCFG() { ompleCampsCFG(llegeixCFG()); }

void mostraCFG() {
  //## Mostra al formulari de frmConfiguració carregada en la memòria
  int iCamps[]={1,2,3,6,4,7,5,8};
  unsigned char i;

  sTmp=config.iPantallaOFF;  escriuLCD(15,sTmp) ; //## Temps salvapantalles
  sTmp=config.iTAlerta;      escriuLCD(18,sTmp) ; //## Ta Avis
  sTmp=config.iLatenciaSMS;  escriuLCD(17,sTmp) ; //## Temps amb alerta aban d'enviar SMS
  sTmp=formatDecimal(config.iCarregadorON,1);  escriuLCD(19,sTmp) ; //## V per connectar carregador

  for (i=0;i<4;i++) { //## Volcat de l'agenda de telèfons
    escriuLCD(iCamps[2*i],agenda.telf[i].numero);
    escriuLCD(iCamps[2*i+1],agenda.telf[i].nom);
  }

  //## IP i mascara del dispositiu
  sTmp=tornaIP();
  if (sTmp=="") sTmp=F("?.?.?.?/?.?.?.?");
  escriuLCD(12, sTmp.substring(0,sTmp.indexOf("/"))); //## Actualitzo IP
  escriuLCD(13, sTmp.substring(sTmp.indexOf("/")+1));   //## Actualitzo Màscara

  //## Adreça web de configuració
  sTmp="http://" + sTmp.substring(0,sTmp.indexOf("/")) + F("tfg/configuracio.php");
  escriuLCD(14, sTmp);   //## Actualitzo adreça configuracio
}
```

```
void ompleCampsCFG(String sAux) {
  //## A partir d'una cadena de text emplena tots els valors de configuracio
  //## Format:  iTempsApagat||T Llindar||Minuts abans SMS||V carregador ON||telf1||Rao1||telf2||Rao2||telf3||Raó3||t
  sAux=tornaProperCamp(sAux); config.iPantallaOFF=sTmp.toInt();
  sAux=tornaProperCamp(sAux); config.iTAlerta=sTmp.toInt(); //## Ta Avis
  sAux=tornaProperCamp(sAux); config.iLatenciaSMS=sTmp.toInt();
  sAux=tornaProperCamp(sAux); config.iCarregadorON=sTmp.toInt();  //## V carregador ON
  for (int i=0;i<4;i++) {
    sAux=tornaProperCamp(sAux); sTmp.toCharArray(agenda.telf[i].numero,12);
    sAux=tornaProperCamp(sAux); sTmp.toCharArray(agenda.telf[i].nom,50);
  }
}

String tornaProperCamp(String sCadena) {
  //## Els camps d'informacio estan entre els caracters || i ||
  int iPos=sCadena.indexOf("||");   //## Trobo el proper separador
  sTmp="";
  if (iPos > 0) sTmp=sCadena.substring(0,iPos);
  else iPos = -2;
  return sCadena.substring(iPos+2);
}

String llegeixCFG() {
  //## Llegeix el contingut de l'arxiu de configuració i el retorna,
  //##    Havent eliminat els caracters d'inici i fi
  //## Format:  iTempsApagat||T Llindar||Minuts abans SMS||V carregador ON||telf1||Ra??telf2||Ra??telf3||Ra??telf4||
  String sAux;
  sAux="config/dadesCFG.php";
  sAux=tornaContingutArxiu(sAux);

  sAux=sAux.substring(sAux.indexOf("/~")+2);     //## El text amb la informació
  sAux=sAux.substring(0,sAux.indexOf("~/"));     //## El text amb la informació

  return sAux;
}


//  ////////////////////////////////////////////
//                      ARXIUS
//  ////////////////////////////////////////////
File tornaFile(String  sSufixArxiu, int Mode_apertura){
  //## Retorna el punter al fitxer desitjat
  sTmp=RUTA_WWW;
  sTmp+=sSufixArxiu;
    return  FileSystem.open(sTmp.c_str(), Mode_apertura);
}

String tornaContingutArxiu(String  sSufixArxiu){
  //## Torna tot el contingut de l'arxiu desitjat
  File oArxiu = tornaFile(sSufixArxiu,FILE_READ);
  sTmp="";
    if (oArxiu) {
    while (oArxiu.available()) {
      sTmp+= (char) oArxiu.read();
    }
    oArxiu.close();
  } else mostraError(F("*** Arxiu NO trobat ***"));
  return sTmp;
}

void escriuFinalArxiu(String  sSufixArxiu, String sContingut) {
  //## Escriu la informacio rebuda al final de l'arxiu indicat
    File oArxiu = tornaFile(sSufixArxiu, FILE_APPEND);
    if (oArxiu) {
      oArxiu.println(sContingut);
      oArxiu.close();
    }  else mostraError(F("No puc escriure al final de l'arxiu"));
}
```

```
//  ////////////////////////////////////////////
//                  ALTRES
//  ////////////////////////////////////////////
String tornaDataHora(char cTipus) {
  //## Retorna la data i/o hora del sistema amb el format demanat
  String sFormat;
  char c;
  Process proc;
  switch (cTipus) {
    case 'D': sFormat="+%Y-%m-%d"; break;
    //case 'H': sFormat="+%H:%M:%S"; break;
    case 'T': sFormat="+%Y-%m-%d %H:%M:%S"; break;
  }
  //delay(100);
  proc.begin("date");
  proc.addParameter(sFormat);
  proc.run();     //## Executa la comanda i espera a que acabi
  sFormat="";

  // read the output of the command
  while(proc.available()>0) {
  c=proc.read();
  if(c!= '\n') sFormat += c;
  }
  return sFormat;
}

String tornaIP() {
    String sMissatge="";
    int i;
    Process p;
    //## S'executa l'ordre d'executar l'escript WifiStatus, (/usr/bin/pretty-wifi-info.lua)
    //## i després es crida a la comanda GREP per tal de quedar-nos només amb la línia
    //## que conté el text  'IP address:'
    p.runShellCommand(F("/usr/bin/pretty-wifi-info.lua | grep 'IP address:'"));
      //## Retorna quelcom semblant a IP address: 192.168.1.120/255.255.255.0

   while (p.running()); //## S'espera a que s'acabi d'executar l'ordre
              //## En acabar tenim tota la cadena retornada

  i=0;
  while (p.available()) {  //## Si s'ha obtingut la informació
    sMissatge+=p.parseInt();
    if(i<4)  { //## cerca els 4 primers números que troba
      sMissatge+=(i<3 ? "." : "/") ;    //## Li posa el punt pel donar format a la IP
    } else {
      if (i<7) sMissatge+=".";
    };
    i++;
    }
   return sMissatge;
}

String formatDecimal(int iDecimal,unsigned char iNumDecimals){
  //## Dona format a un enter com si fos un decimal
  float fDec=iDecimal;
  for (int i=0;i<iNumDecimals;i++) fDec/=10;
  sTmp= fDec;
  return sTmp.substring(0,sTmp.indexOf(".") + 1 + iNumDecimals);
}

boolean latenciaSuperada(unsigned long ulIniciAlerta) {
  return (millis() > ulIniciAlerta + config.iLatenciaSMS * 60000L);
}

unsigned long tornaMarcaTemps(unsigned int uiSegons){
  return millis() + (uiSegons * 1000L);
}
```

```
350 ////////////////////////////////////////////////////////////////////
351 //                      SENSORS
352 // ////////////////////////////////////////////////////////////////
353
354 void mostrejaSensors() {
355   //## El sensor de temperatura és lineal amn la temperatura i per
356   //##  cada grau d'augment la sortida s'incrementa 10mv
357   //## Per estalviar memòria treballo només amb un dígit decimal i ho mulitplico
358   //##  per 10 per emmagatzemar-ho com a integer
359   String sAux="";
360   accions.ulProperMostreig=tornaMarcaTemps(5);  //## Un  mostreig cada 5 segons
361
362   lecturaSensors();
363   actualitzaPantallaEstat();
364
365   if (millis() >= accions.ulProperLog) {
366     //## Si és el moment, desa el registre a l'arxiu
367     sAux="logs/tfg_";
368     sAux+=tornaDataHora('D')+".txt";
369     escriuFinalArxiu(sAux, tornaEstat());
370     accions.ulProperLog=accions.ulProperMostreig + 55000L; //## Un  mostreig cada 60 segons (55+5)
371   }
372 }
373
374 void lecturaSensors() {
375   //## Totes les lectures es desen com a integers per estalviar espai de memòria
376   estat.iTempS = ((5.0 * lecturaAnalogica(SENSOR_T1) * 100.0) )*10;  //## La mostra llegida es converteix a temperat
377   estat.iTempAC = ((5.0 * lecturaAnalogica(SENSOR_T2) * 100.0) )*10;  //## La mostra llegida es converteix a tempera
378   estat.bAireCondicionat_ON = (digitalRead(SENSOR_REED)== HIGH ? true : false );
379   estat.iV2= ((V2 * lecturaAnalogica(SENSOR_TENSIO)) ) * 100.0 ;  //## Tensió de les bateries, passada pel divisor,
380                               //## la llegida és un 0.68% de real.
381   estat.bAmbDC=(digitalRead(SENSOR_AC)== HIGH  ? false : true);
382 }
383
384 float lecturaAnalogica( unsigned char ucPin ){
385   return analogRead(ucPin) / 1024.0;
386 }
387
388 unsigned int tornaValorGrafic(unsigned int uiValor) {
389   //## L'alçada del gràfic és de 165 pixels
390   //## Fixo la temperatura màxima a 35 graus per tal que no surti de la gràfica
391   //## Desplaço l'escala 10 graus per tal de començar a 10 graus
392   //## El marge de temperatures és de 25 graus (35 - 10)
393   //## El gràfic està magnificat a un 100% i per això els valors van multiplicats per 10
394   return ((uiValor>350 ? 350 : uiValor)-100)/250.0 * 165 ;
395 }
396
397 void actualitzaPantallaEstat() {
398   //## Display i gràfic temperatura sala
399   oLCD.WriteObject(GENIE_OBJ_LED_DIGITS, 0x00, estat.iTempS);     //## L'objecte de la pantalla no entén els decimal
400   oLCD.WriteObject(GENIE_OBJ_SCOPE, 0x00, tornaValorGrafic(estat.iTempS)); //## L'objecte de la pantalla va canviant
401                                     //## Per evitar que dibuixi fora del gràfic
402
403   //## Display i gràfic temperatura AC
404   oLCD.WriteObject(GENIE_OBJ_LED_DIGITS, 0x01, estat.iTempAC);    //## L'objecte de la pantalla no entén els decimal
405   oLCD.WriteObject(GENIE_OBJ_SCOPE, 0x00, tornaValorGrafic(estat.iTempAC)); //## L'objecte de la pantalla va canvian
406
407   //## Led comporta AC oberta
408   oLCD.WriteObject(GENIE_OBJ_USER_LED, 0x00, estat.bAireCondicionat_ON);
409
410   //## Display tensió bateries
411   oLCD.WriteObject(GENIE_OBJ_LED_DIGITS, 0x02, estat.iV2 );     //## Actualitza valor tensió
412
413   //## LEDS funcionament AC/DC
414   oLCD.WriteObject(GENIE_OBJ_LED, 0x00, !estat.bAmbDC*1 );    //## Actualitza led funcionant amb AC
415   oLCD.WriteObject(GENIE_OBJ_LED, 0x01, estat.bAmbDC*1 );     //## Actualitza led funcionant amb DC
416
417   //## LED  carregador encés
418   oLCD.WriteObject(GENIE_OBJ_USER_LED, 0x01, estat.bCarregant);
419
420
421   //## Verificació alerta per excés de temperatura
422   if (estat.iTempS>config.iTAlerta*10) {
423     if (!estat.bAlertaT) {
424       estat.bAlertaT=true;
425       accions.ulIniciAlertaT=millis();
426       mostraError(F("Temperatura per sobre el llindar"));
427     }
428   } else if (estat.bAlertaT) { estat.bAlertaT=false; }
429
```

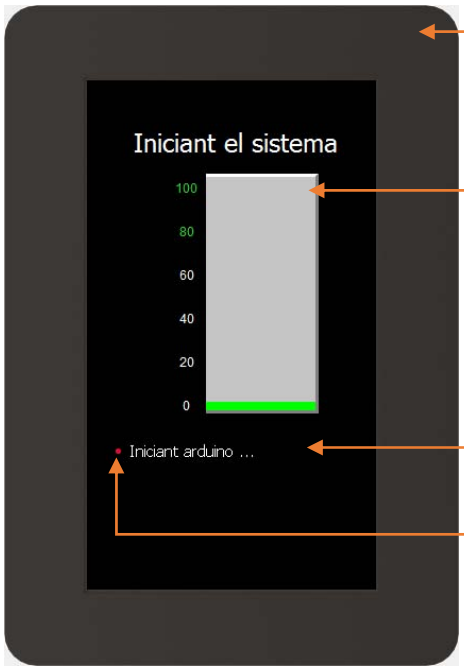
```
  if (estat.bAlertaT && latenciaSuperada(accions.ulIniciAlertaT) ) {
    accions.ulIniciAlertaT=millis();
    enviaSMS(1);
  }

    //## Verificació alerta per funcionament amb DC
  if (estat.bAmbDC ) {
    if (!estat.bAlertaDC) {
      estat.bAlertaDC=true;
      accions.ulIniciAlertaDC=millis();
      mostraError(F("Funcionant amb bateries"));
    }
  } else if (estat.bAlertaDC) {
    estat.bAlertaDC=false;
    if (latenciaSuperada(accions.ulIniciAlertaDC) ) {
      enviaSMS(3);
    }
  }
  if (estat.bAlertaDC && latenciaSuperada(accions.ulIniciAlertaDC)) {
    accions.ulIniciAlertaDC=millis(); //## Reinicio comptador enviament SMS
    enviaSMS(2);
  }

   //## Verificació si cal engegar carregador bateries
   if (estat.iV2 < (config.iCarregadorON-config.iCarregadorON) ) {
    if (!estat.bCarregant) {
      activaCarregador(true);
      accions.ulFiCarrega=tornaMarcaTemps(18000); //## 5h carregant (300min * 60s);
    } else if (millis()>accions.ulFiCarrega){
      activaCarregador(false);
    }
  } else if (estat.bCarregant && (millis()>accions.ulFiCarrega) ) {
      activaCarregador(false);
    }
}

String tornaEstat() {
  //## Retorna una cadena amb la infromació dels sensors separada per tabuladors
  String sAux=tornaDataHora('T')+"\t";
  sAux+=formatDecimal(estat.iTempS,1)+"\t";
  sAux+=formatDecimal(estat.iTempAC,1)+"\t";
  sAux+=(estat.bAireCondicionat_ON ? "1\t" : "0\t");
  sAux+=formatDecimal(estat.iV2,2)+"\t";
  sAux+=(estat.bCarregant ? "1\t" : "0\t");

  return sAux;
}

void activaCarregador(boolean bActiva) {
  if (bActiva) {
    digitalWrite(RELES_1_2, LOW); //## Primer desconnecta de la xarxa les bateries
    delay(1000);
    digitalWrite(RELE_4, LOW);    //## i després connecta el carregador
  } else {
    digitalWrite(RELE_4, HIGH);   //## Primer desconnecta el carregador
    delay(1000);
    digitalWrite(RELES_1_2, HIGH);  //## i després connecta les bateries al circuit
  }
  estat.bCarregant=bActiva;
}



////////////////////////////////////////////////////////////////////////
//                     MÒDUL PANTALLA LCD
// ////////////////////////////////////////////////////////////////////
void escriuLCD(int iIndex, String sCadena) {  oLCD.WriteStr(iIndex,sCadena);  }

void salvaPantalla() {

   oLCD.WriteObject(GENIE_OBJ_FORM, 0x03, 1); //## Activa formulari de l'LCD
                       //## que té botó que ocupa tota la pantalla
                                          //## Per tal que es reactivi la pantalla en tocar-la
   oLCD.WriteContrast(0);
}

```

```
506
507 void mostraError(String sCadenaError){
508   String sAux=F("Incidència detectada!");
509   oLCD.WriteObject(GENIE_OBJ_FORM , 0x04, 1); //## Activa formulari per mostrar missatges
510
511   escriuLCD(16,sAux.c_str());
512   escriuLCD(10,sCadenaError.c_str());
513
514   estat.bMostrarError=true;
515 }
516
517
518
519 // ---------------------------------------------------------------
520 //
521 // This is the user's event handler. It is called by oLCDDoEvents()
522 // when the following conditions are true
523 //
524 //    The link is in an IDLE state, and
525 //    There is an event to handle
526 //
527 // The event can be either a REPORT_EVENT frame sent asynchronously
528 // from the display or a REPORT_OBJ frame sent by the display in
529 // response to a READ_OBJ request.
530 //
531 // ---------------------------------------------------------------
532
533 void myGenieEventHandler(void) {
534   genieFrame Event;
535   oLCD.DequeueEvent(&Event);
536   int iIndex;
537   accions.ulPantallaOFF = tornaMarcaTemps(config.iPantallaOFF); //## Actualitzo el temps
538             //## d'apagat de pantalla per a qualsevol event que
539           //##vingui de la pantalla
540
541   if(Event.reportObject.cmd == GENIE_REPORT_EVENT) {
542   iIndex=Event.reportObject.index;
543     switch(Event.reportObject.object) {
544     case GENIE_OBJ_WINBUTTON: //## Hi ha un event WINBUTTON pendent de gesitonar
545       gestionaEvent_winButton(iIndex);
546       break;
547     case GENIE_OBJ_USERBUTTON: //## Hi ha un event USERBUTTON pendent de gesitonar
548       switch(iIndex)  {
549         case  8:  //## Pitjat botó ocult a frmIniciant
550           oLCD.WriteContrast(15);
551           oLCD.WriteObject(GENIE_OBJ_FORM, 0x00, 1);
552           break;
553       }
554       break;
555     case GENIE_OBJ_FORM:    //## Hi ha un event FORM pendent de gesitonar
556       switch(iIndex)  {
557         case  2: //## Entrada a formulari frmConfiguracions
558           mostraCFG();
559         break;
560       }
561       break;
562   }
563   }
564 }
565
```

```
void gestionaEvent_winButton(int iIndex) {
  String sAux;
  switch(iIndex)  {
    case 0: //## Pitjat botó Comprovacions a frmEstat
      oLCD.WriteObject(GENIE_OBJ_FORM , 0x06, 1);   //## Activa formulari de l'LCD que té un botó que  ocupa tota la
      break;
    case 4: //## Pitjat botó envia SMS a frmComprovacions
      oLCD.WriteObject(GENIE_OBJ_FORM , 0x04, 1); //## Activa formulari per mostrar missatges
      enviaSMS(1);   //## Envia SMS de prova
      break;
    case 6: //## Pitjat botó tanca pantalla a frmMenu
      salvaPantalla();
      break;
    case 7: //## Pitjat botó tanca pantalla a frmComprovacions
      activaCarregador(!estat.bCarregant);
      break;
    case 9: //## Pitjat el botó conforme s'ha llegit avís a frmMissatges
      oLCD.WriteObject(GENIE_OBJ_FORM , 0x00, 1);   //## Activa formulari d'estat
      estat.bMostrarError=false;
      break;
    case 10:  //## Pitjat botó tanca pantalla a frmMenu
      encenSIM900();
      break;

  }
}



void resetPantalla() {
  //Reset the Display (change D4 to D2 if you have original 4D Arduino Adaptor)
  //See the app note 4D-AN-P4017-ViSi-Genie Connec ting a 4D Display to an Arduino Host
  //for information if using jumper connecting wires

 // THIS IS IMPORTANT AND CAN PREVENT OUT OF SYNC ISSUES, SLOW SPEED RESPONSE ETC
  // If NOT using a 4D Arduino Adaptor, digitalWrites must be reversed as Display Reset is Active Low, and
  // the 4D Arduino Adaptors invert this signal so must be Active High.

  pinMode(RESET_P, OUTPUT);  // Set D4 on Arduino to Output (4D Arduino Adaptor V2 - Display Reset)
  digitalWrite(RESET_P, 1);  // Reset the Display via D4
  delay(100);
  digitalWrite(RESET_P, 0);  // unReset the Display via D4

  delay (3500);         //let the display start up
}

//////////////////////////////////////////////////////////////////////////
//                    MÒDUL GSM
//----------------------------------------------------------------------
void encenSIM900() {
  //## Encesa/tancada per software. és equivalent a pitjar el botó d'encesa
  //##  a l'escut GSM
  estat.bGSM_ON=!estat.bGSM_ON;
  digitalWrite(SIM900_OnOff, HIGH);
  delay(1000);  //## Ha d'estar un segon en estat alt
  digitalWrite(SIM900_OnOff, LOW);
  delay(3000);  //## Cal esperar un temps abans que hi ha dades al buffer
}
```

```
void enviaSMS(char tipus)  {
  //## Funció per enviar el missatges de SMS
 String sAux="";
 sTmp=tornaContingutArxiu((tipus==1 ? F("config/sms_text.txt") : (tipus==2 ? F("config/sms_dc.txt") : F("config/sms_

 escriuLCD(10, F("Enviant SMS \n"));
 escriuLCD(10, sTmp);

  for (int i=0;i<2;i++) {
    delay(500);
    oSIM900.print(F("AT+CMGF=1\r")); // AT command to send SMS message
    delay(1000);
    sAux="AT+CMGS=\"";
    sAux+=agenda.telf[i].numero;
    sAux+="\"";
    oSIM900.println(sAux);  //##Número de telefon
    delay(1000);
    oSIM900.println(sTmp);  //## Text del SMS
    delay(100);
    oSIM900.println((char)26); // End AT command with a ^Z, ASCII code 26 //Comando de finalizacion
    delay(100);
    oSIM900.println();
    delay(5000);    //##  Temps d'espera per tal que es garanteixi l'enviament del missatge
    oSIM900.flush();
  }
  escriuLCD(10, F("SMS enviat \n"));

}
//  ////////////////////////////////////////////
//                     SERVIDOR WEB
//  ////////////////////////////////////////////
void servidorWeb() {
  String peticio;
  BridgeClient client = oHTTP.accept(); //##Accepta peticions del servirdor web
  if (client) {             //## Si hi arrba una nova petició ...
    peticio = client.readStringUntil('/');
    peticio.trim();        //## Treu possibles espais en blanc

    sTmp=F("Comanda desconeguda ");
    sTmp+=peticio ;
    if (peticio == F("tfg_info")) { sTmp=tornaEstat(); }
    else if (peticio == F("tfg_repCFG")) {
        carregaCFG(); //## Omplo estructura config
        sTmp=F("Dispositiu reconfigurat");
    }

    client.println(sTmp.c_str());
    client.stop();            //## Tanca la connexió
  }

}
```

Annex 6.6 Pantalles i objectes del sistema creats en l'entorn ViSi Genie del Workshop 4 IDE de 4D Systems



Form 5

Tank 0

Strings 9

Strings 11

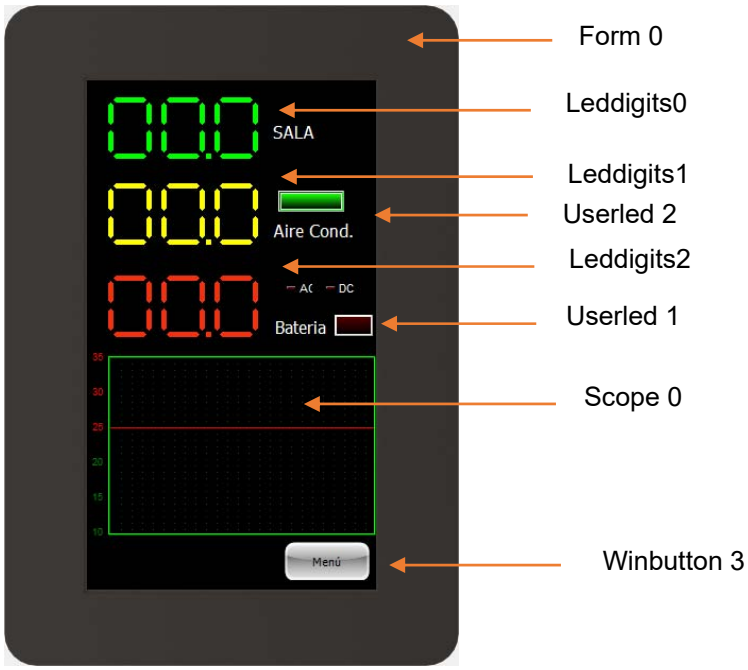
Form 1

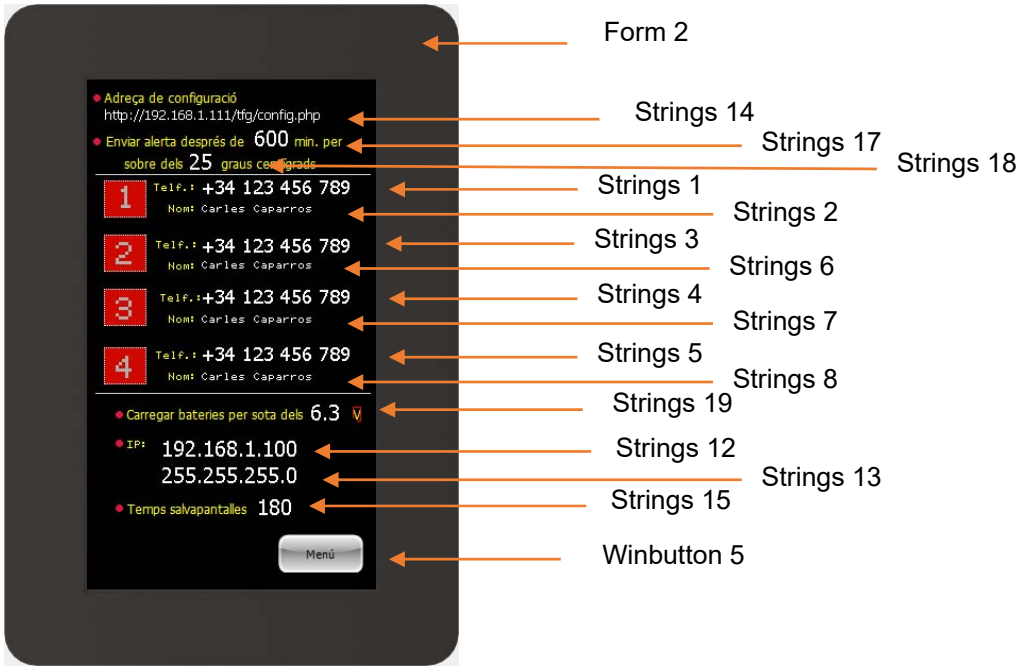
Winbutton 1

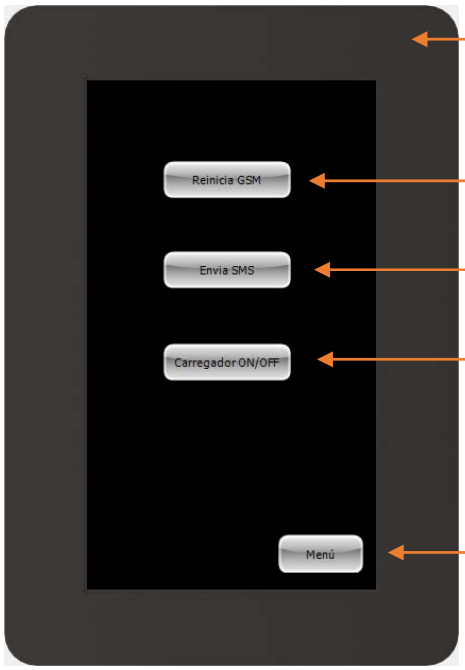
Winbutton 2

Winbutton 0

Winbutton 6


SALA
Aire Cond.
AC
DC
Bateria
Menú
Form 0
Leddigits0
Leddigits1
Userled 2
Leddigits2
Userled 1
Scope 0
Winbutton 3


Adreça de configuració
http://192.168.1.111/tfg/config.php
Enviar alerta després de 600 min. per
sobre dels 25 graus centígrads
1 Telf.: +34 123 456 789
Nom: Carles Caparros
2 Telf.: +34 123 456 789
Nom: Carles Caparros
3 Telf.: +34 123 456 789
Nom: Carles Caparros
4 Telf.: +34 123 456 789
Nom: Carles Caparros
Carregar bateries per sota dels 6.3 V
IP: 192.168.1.100
255.255.255.0
Temps salvapantalles 180
Menú
Form 2
Strings 14
Strings 17
Strings 18
Strings 1
Strings 2
Strings 3
Strings 6
Strings 4
Strings 7
Strings 5
Strings 8
Strings 19
Strings 12
Strings 13
Strings 15
Winbutton 5



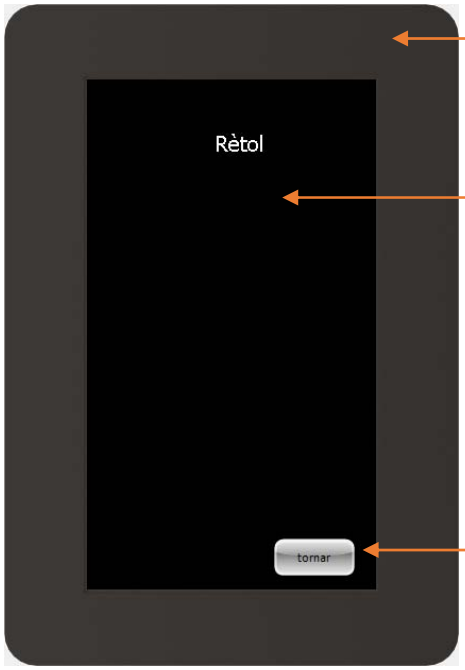
Form 6

Winbutton 10

Winbutton 4

Winbutton 7

Winbutton 8



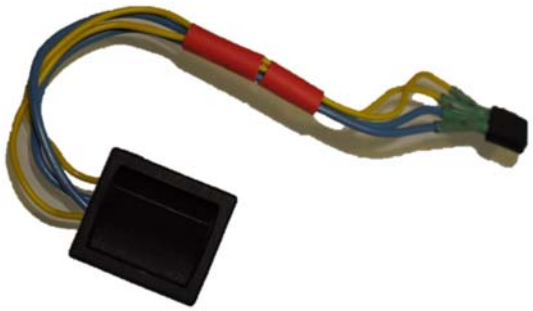
Form 4

String 10

Winbutton 9

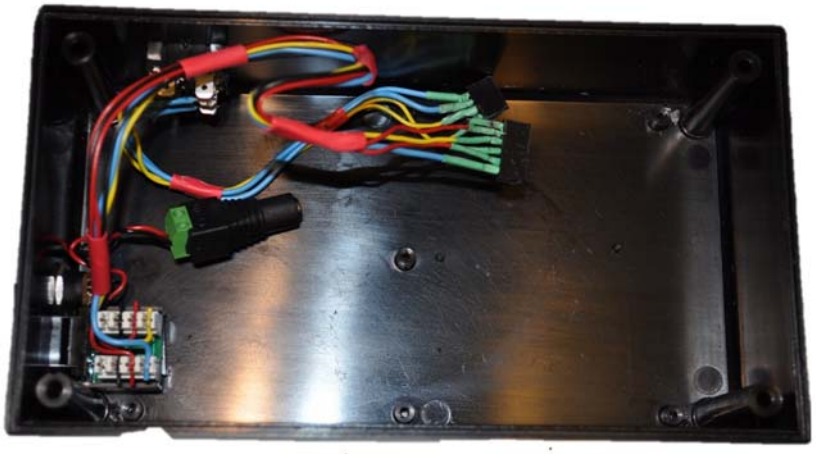
Annex 6.7 Fotos del muntatge del projecte

Interruptor general



Connector cap els sensor de l'aire condicionat

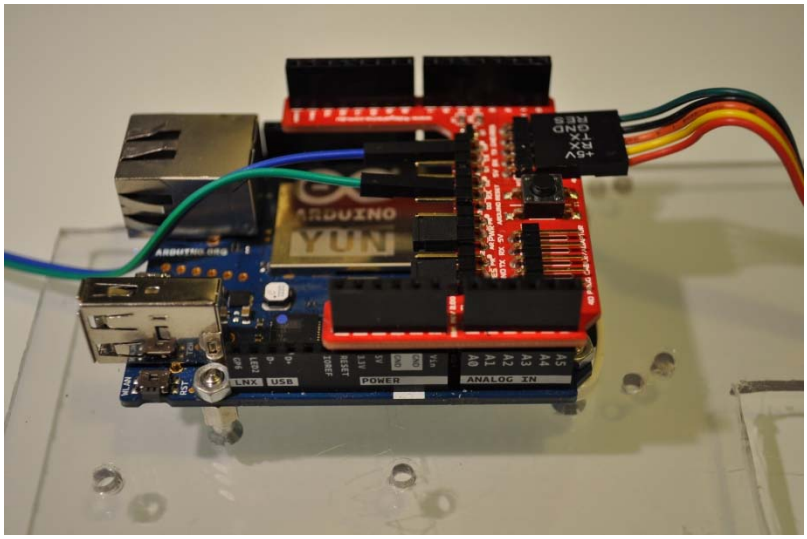
Connectors dins la caixa



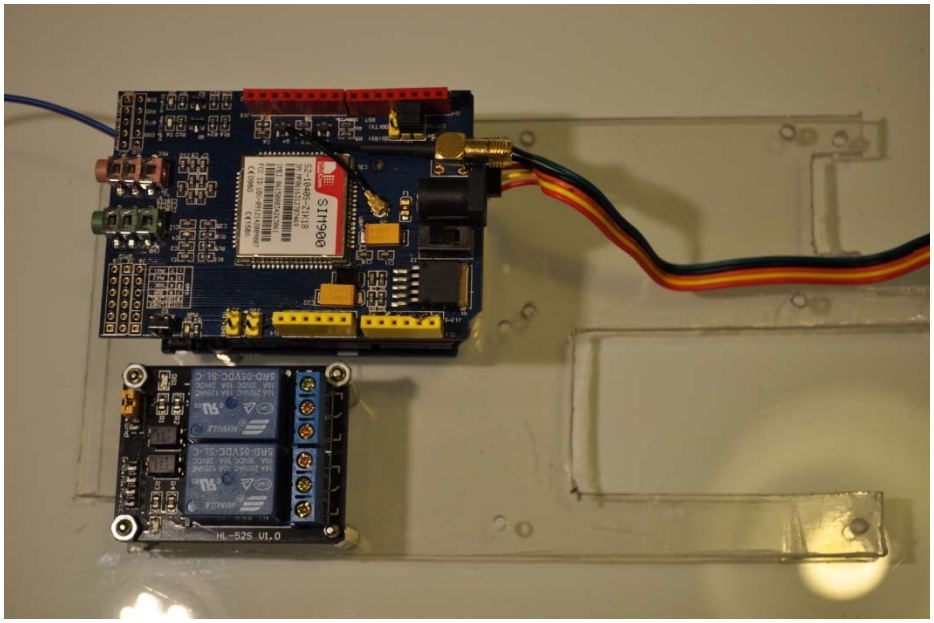
Placa de plàstic on es fixen tots el components

Arduino

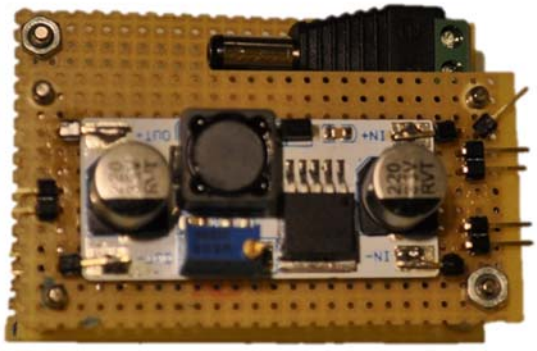
Escut de la pantalla



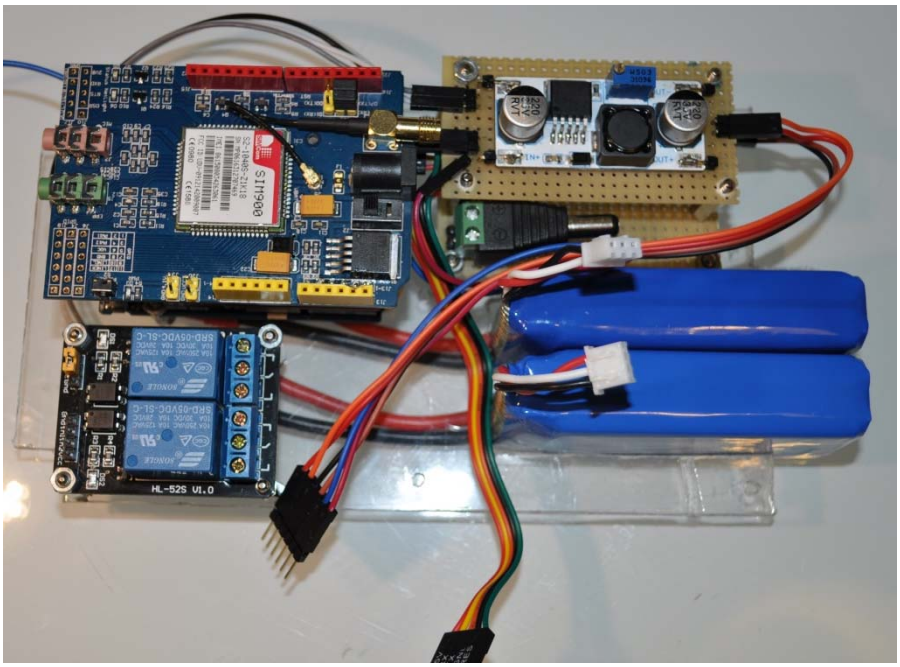
Escut GSM i relés



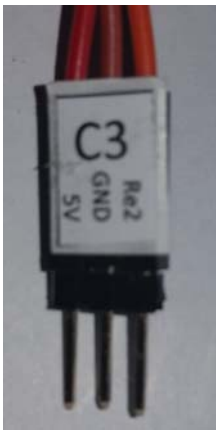
Suport per al mòdul DC-DC i entrada de l'alimentació provinent de la xarxa elèctrica

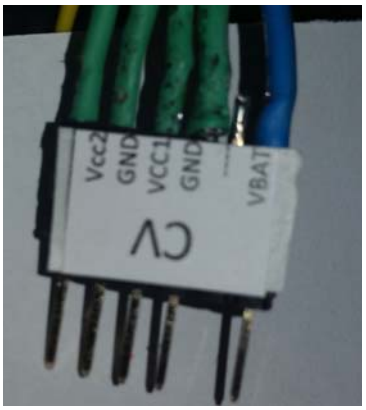


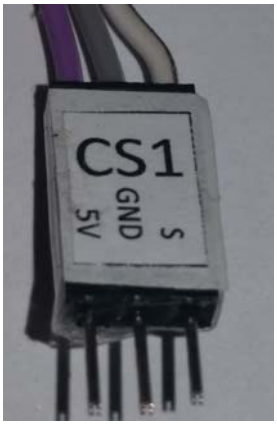
Tot preparat per encabir-ho dins la caixa

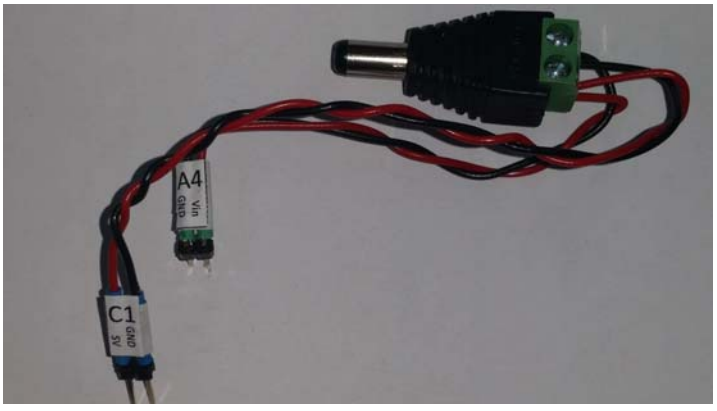


Connectors etiquetats

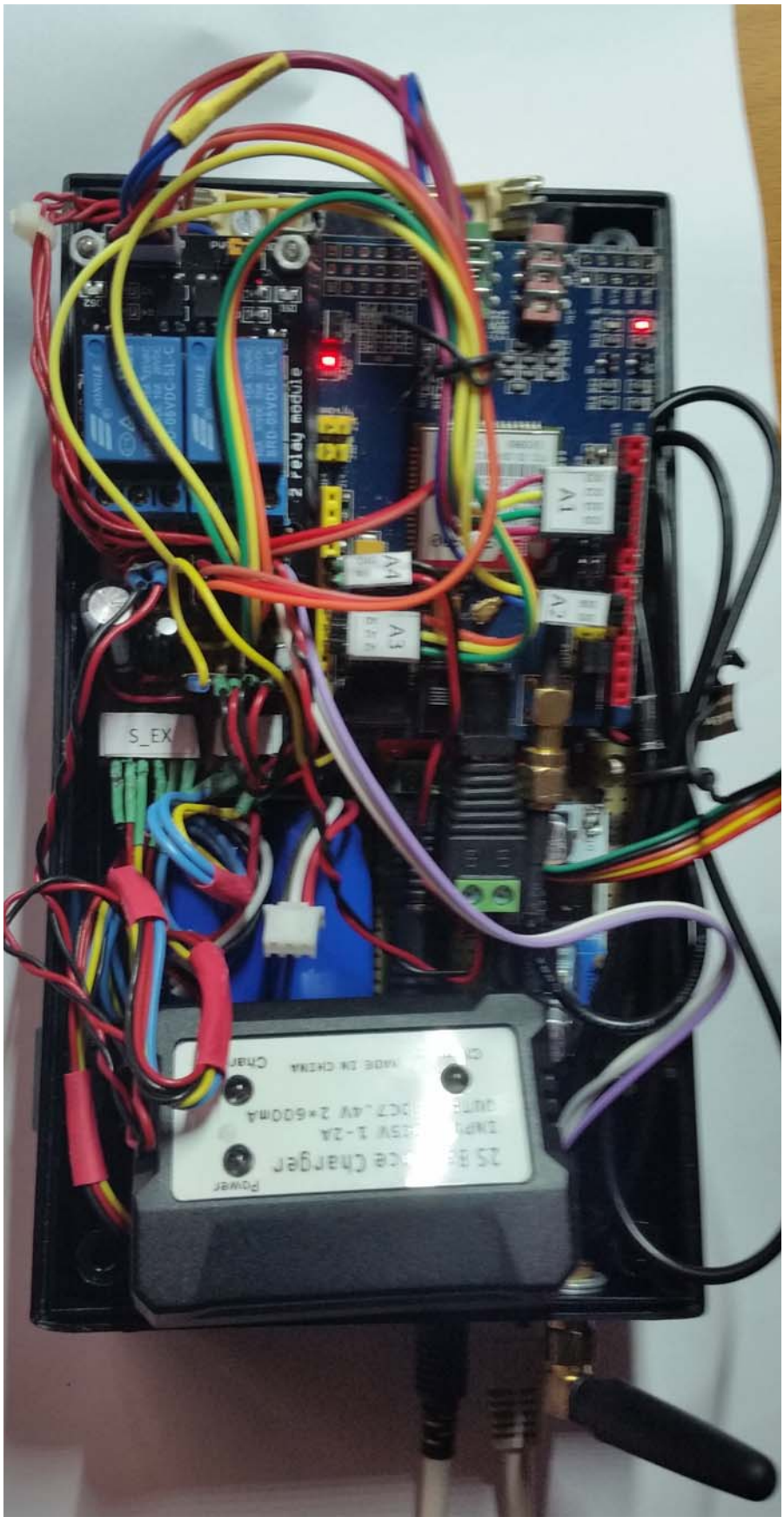








Tot preparat per tancar



Els sensors de l’aire condicionat al seu lloc