EPS
Escola Politècnica Superior
UdG

Projecte/Treball Fi de Carrera

Estudi: Grau en Enginyeria Electrònica Industrial i Automàtica

Títol: Disseny d'un node Art-Net amb dues sortides DMX

Document: 1. Memòria

Alumne: Xavier Herranz Izquierdo

Director/Tutor: Daniel Alexandre Macaya Masferrer
Departament: Enginyeria Elèctrica, Electrònica i Automàtica
Àrea: ESA

Convocatòria (mes/any): febrer/2014

Índex

1. INTRODUCCIÓ

1.1. Antecedents

DMX512 és un estàndard de comunicació digital, fet servir principalment en luminotècnica, que consisteix en la transmissió de comandes mitjançant canals d'un byte cadascun. Cada trama DMX512 pot arribar a tenir fins a 512 canals. Les característiques elèctriques del senyal queden definides per l'estàndard RS-485.

El protocol Art-Net permet enviar trames del protocol DMX512 fent servir datagrames UDP, que es poden transmetre mitjançant una xarxa Ethernet de qualsevol tipus de topologia.

L'ús d'Ethernet per transmetre les dades DMX512 comporta varis avantatges. Primerament redueix considerablement el cost del cablejat i de l'equipament de l'instal·lació. Un sol cable Cat5 (a 100Mbits/s) permet transportar fins a 400 universos, quan un cable XLR només pot transportar-ne un. Generalment, també, l'equipament típic de xarxes Ethernet (hubs, routers) és molt més econòmic i comú que el destinat a xarxes DMX512.

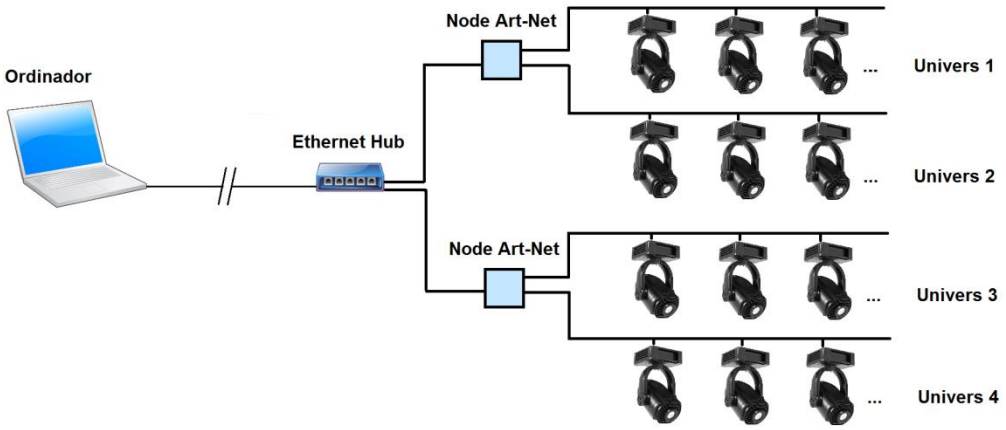

Figura 1. Distribució de quatre universos DMX512 amb Art-Net

La utilització cada cop més freqüent de matrius de LEDs, per exemple, ha comportat un increment molt important del número d'universos necessaris per a una única instal·lació. És en situacions com aquesta on la capacitat del protocol Art-Net de transmetre fàcilment nombrosos universos llueix més.

Figura 2. Aplicació d'Art-Net a l'hotel Puerta de Amèrica, Madrid

Un altre avantatge es troba en un potencial increment de l'abast de la xarxa. Tot i que la llargada màxima d'un cable CAT5 és menor a la d'un cable XLR, el cost molt menor dels hubs Ethernet nega aquesta diferència. També existeix la possibilitat d'emprar Wi-Fi o fins i tot transmetre les comandes via Internet amb una VPN (Virtual Private Network).

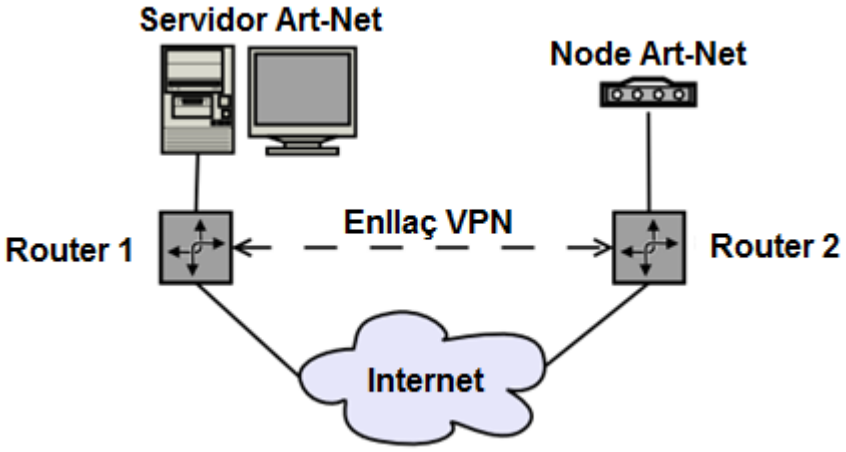

Figura 3. Xarxa Art-Net via Internet

Per tal de comandar els dispositius que fan servir el protocol DMX512, farà falta un node que interpreti els missatges Art-Net i extregui els universos (trama DMX512 de 512 canals) que contenen. Es podria dir que el node Art-Net actuarà com a traductor entre la xarxa Ethernet i la xarxa DMX512.

1.2. Objecte

L'objecte del projecte es dissenyar un dispositiu capaç de transmetre dos universos DMX512 a partir de les dades rebudes fent servir el protocol Art-Net.

El node haurà de ser capaç de canviar la seva configuració ja sigui mitjançant missatges propis del protocol Art-Net com mitjançant una plana web accessible des de la adreça IP del node.

1.3. Abast

Es dissenyarà el circuit electrònic del node, que inclou la lògica de control del node, els mitjans d'entrada i sortida dels senyals Art-Net i DMX512, els elements de interfície d'usuari, l'alimentació del conjunt, i el programari corresponent.

A nivell del protocol, el node complirà amb totes les especificacions relatives als missatges i la transmissió d'aquests a excepció de les funcionalitats RDM (recepció de senyals DMX512) i les d'actualització del firmware.

Finalment s'elaborarà un prototip del disseny per tal de demostrar el seu funcionament.

2. EL PROTOCOL ART-NET

Desenvolupat per la companyia Artistic Licence Engineering Ltd, Art-Net és un estàndard obert que permet la transmissió de trames de l'estàndard DMX512 a través del protocol UDP.

Actualment es troba en la seva tercera versió, Art-Net 3, que ofereix sobre les anteriors la capacitat de suportar un número molt més elevat de universos, fins a 32.768.

2.1. Especificacions de l'adreçament

Els missatges del protocol Art-Net s'envien fent servir el protocol UDP. Cada node ha de disposar d'una IP única, que es pot atorgar de forma estàtica o fent servir un servidor DHCP. El port de destí i recepció dels paquets UDP es el mateix, el 6454.

L'adreça IP de cada node ha de ésser de tipus A, es a dir, amb la màscara de subxarxa 255.0.0.0, i l'adreça ha de tenir la forma de 2.0.0.0/8. L'especificació també accepta adreces amb la forma 10.0.0.0/8.

Els paquets poden ser enviats mitjançant adreçament broadcast o unicast. En funció d'aquesta elecció, el número d'universos que es poden enviar al mateix temps varia. Per no sobrecarregar la xarxa amb missatges es recomana enviar els paquets amb adreçament unicast.

	Ethernet 10BaseT	**Ethernet 100BaseT**	**Ethernet 1000BaseT**
Broadcast	40	40	40
Unicast	40	400	4000+

Taula 1. Número màxim recomanat de universos per xarxa

2.1.1. Adreçament de l'univers DMX512

Cada port de sortida o entrada DMX512 del node Art-Net té una adreça de 15 bits que es composa de tres camps: la xarxa, la subxarxa i l'univers. Cada node només pot processar paquets DMX d'una única xarxa i subxarxa, i pot arribar a tenir 4 universos de sortida o entrada diferents.

Bit 15	Bits 14-8	Bits 7-4	Bits 3-0
0	Xarxa	Subxarxa	Univers

Taula 2. Estructura de l'adreça del port de sortida DMX512

2.2. Tipus de paquets

El protocol disposa de varis tipus de paquets en els que es basa la comunicació entre nodes de la xarxa. Cada paquet es precedit per un camp de 8 bytes que conté l'array "Art-net/0", el codi del tipus de paquet del missatge i el número de versió del protocol. El missatge es rebutjat si aquesta seqüència inicial no es duta a terme de forma correcta o les versions del protocol difereixen.

El format dels camps dels paquets s'especifiquen de forma similar als tipus de variables de llenguatges de programació d'estil C, tals com INT8 i INT16. Aquesta característica facilita la gestió de rebre i processar els missatges.

Les descripcions de paquets que segueixen tenen per objecte només aquells paquets implementats al node dissenyat, i no representen la totalitat del protocol.

2.2.1. ArtPoll

Missatge destinat al descobriment de nodes Art-Net a la xarxa. El missatge s'envia a la direcció broadcast de la subxarxa i tots els nodes d'aquesta han de respondre amb un missatge de tipus ArtPollReply. També es capaç de configurar el comportament dels nodes de la xarxa a l'hora d'enviar missatges de tipus ArtPollReply, per exemple, forçant-los a enviar-ne un si la seva configuració ha estat canviada.

2.2.2. ArtPollReply

Paquet enviat pels nodes per tal de donar-se a conèixer en la xarxa. Aquest paquet es enviat també a la direcció de broadcast de la subxarxa. Conté tota mena d'informació sobre el node, tal com el tipus de node que és (entrada o sortida de DMX), el seu estat actual o els universos que transmet/rep.

2.2.3. ArtIpProg

Paquet que permet canviar l'adreça IP d'un node, la màscara de subxarxa que fa servir o el port UDP. També es capaç d'habilitar la funcionalitat DHCP del node si és disponible. Un cop s'han efectuat els canvis, el node ha de respondre amb un paquet de tipus ArtIpProgReply.

2.2.4. ArtIpProgReply

Paquet de resposta al paquet ArtIpProg. Conté la IP i la màscara de subxarxa del node que ha estat objecte de la modificació. També indica si la funcionalitat DHCP s'ha activat.

2.2.5. ArtAddress

Paquet destinat a canviar la configuració dels ports del node destí (universos d'entrada i sortida) i també pot configurar el mètode d'unió de les trames DMX512 en cas de que més d'una IP s'estigui adreçant al mateix port del node. Mitjançant aquest paquet també es pot deshabilitar l'unió de trames DMX512 procedents de paquets ArtDMX enviats per varies IP.

Un cop el node hagi efectuat els canvis establerts al paquet, haurà de respondre amb un paquet de tipus ArtPollReply amb l'informació actualitzada.

2.2.6. ArtDMX

Paquet bàsic del protocol, conté els canals DMX512 i l'univers al que pertanyen.

2.3. Programari

Existeix un ampli ventall de programes capaços de generar un senyal de sortida d'Art-Net, tals com Freestyler, Lightjams o Madrix. Aquests programes estan preparats per controlar tota mena de dispositius compatibles amb DMX512, disposant de bases de dades amb els dispositius i els canals que fan servir per a cada funció.

També poden permetre la creació dinàmica d'espectacles luminotècnics en funció de, per exemple, pistes musicals, vídeos o seqüències MIDI.

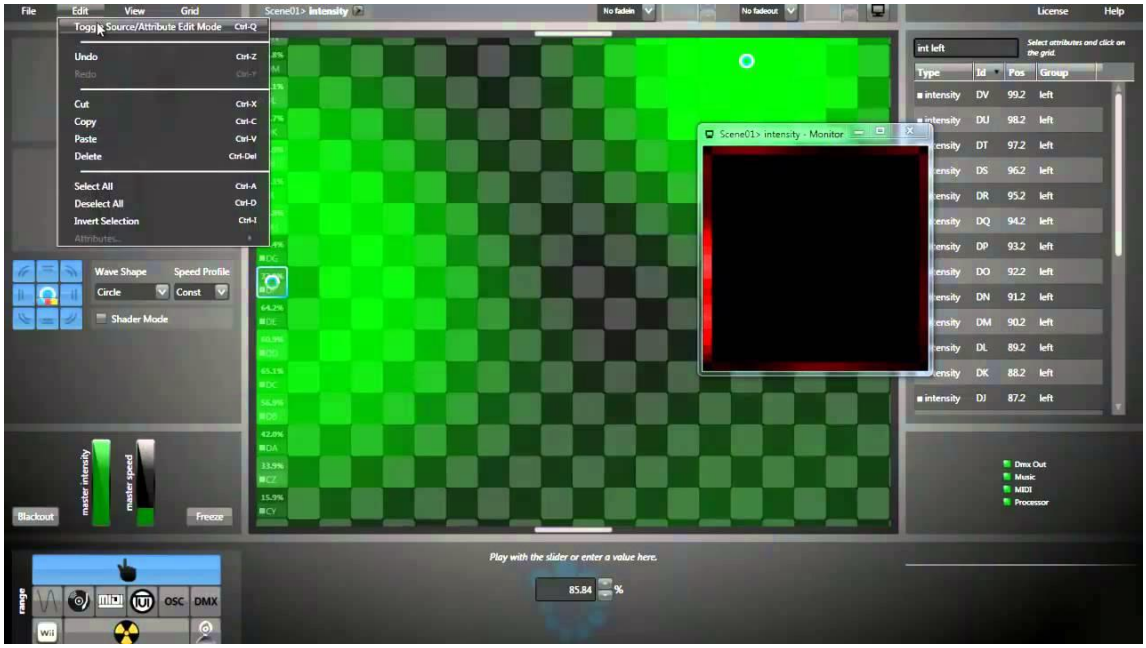

Figura 4. Ús de l'aplicació Lightjams

3. L'ESTÀNDARD DMX512

DMX512 és un estàndard de comunicació digital de dades, fet servir en equips de luminotècnia principalment.

3.1. Característiques elèctriques del senyal

Les dades DMX512 són transmeses mitjançant l'estàndard EIA-485 sobre cable de parell trenat. Dos conductors són utilitzats per enviar el senyal, essent el senyal de tipus balancejat. Aquesta característica garanteix certa immunitat del senyal al soroll, permetent cubrir distàncies majors.

Les xarxes DMX512 tenen una topologia de tipus bus, amb un màxim de 32 dispositus connectats simultàniament, i una longitud de com a màxim 1.200 metres. Es necessari instal·lar resistències de terminació de 120Ω al final del bus per evitar reflexions del senyal.

3.1.1. Connectors

L'especificació utilitza connectors elèctrics de tipus XLR de 5 pins. En el cas que el connector rebi senyals DMX512, el connector serà mascle, i si el port transmet DMX512 haurà de ser femella. Els dos pins no utilitzats es poden fer servir per transmetre un segon senyal.

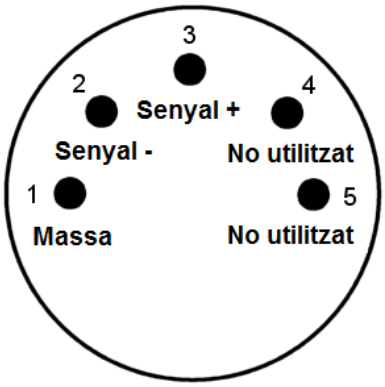

Figura 5. Distribució dels pins al connector XLR

3.2. El protocol

Les dades a l'estàndard DMX512 són transmeses en sèrie i a 250.000 bits/s de forma asíncrona. El format de les dades es composa d'un bit d'start, vuit bits de dades i dos bits d'stop. No s'afegeixen bits de paritat.

3.2.1. Temporització del senyal

Cada paquet comença amb un espai de break seguit per un espai anomenat marca després del break o MAB (Mark After Break). Després ja s'inicia la transmissió de dades del paquet, que pot arribar a contenir 512 canals, d'un byte cadascun.

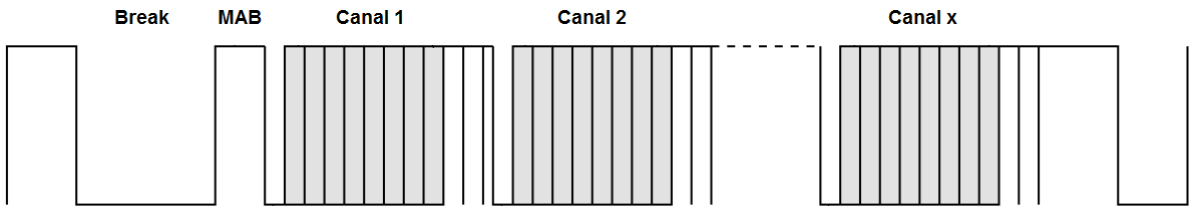

Figura 6. Diagrama de temps del senyal DMX512

Descripció	Minim (µs)	Típic (µs)	Màxim (µs)
Break	92	176	-
MAB	12	-	10^6
Bit	3.92	4	4.08

Taula 3. Temps del senyal DMX512

4. HARDWARE DEL NODE ART-NET

El projecte ha requerit el disseny d'un circuit específic per poder satisfer les necessitats del node. L'element principal d'aquest disseny es un microcontrolador, que gestionarà els missatges Art-Net, l'informació a enviar i rebre de la plana web de configuració, i generarà el parell de senyals de sortida DMX.

La comunicació amb la xarxa Ethernet i la gestió dels missatges del protocol TCP/IP recau sobre el xip W5100, i l'adaptació del senyal de sortida del canal USART a l'estàndard RS485 la realitzarà el xip SN75176.

4.1. Especificacions del microcontrolador

El microcontrolador escollit per al projecte ha estat l'ATmega1284p de la casa Atmel. Les seves característiques més significatives pel projecte es resumeixen a la taula següent:

Encapsulat PDIP 40 pins
128K Bytes de memòria de
16K Bytes de memòria SRAM
4K Bytes de memòria EEPROM
2 ports programables USART
Interfície SPI Master/Slave
2 Timers de 16 bits
14 mA de consum a 16MHz

Taula 4. Especificacions del microcontrolador

El microcontrolador treballarà a una freqüència de 16MHz. L'elecció d'aquest valor es basa en que es el valor més elevat, dins el marge de freqüències recomanades pel fabricant, que permet generar senyals de sortida al canal USART a 250.000kHz (freqüència feta servir per la transmissió de trames del protocol DMX512) amb un màxim de precisió.

4.1.1. Justificació de l'elecció

El principal motiu pel qual es va escollir un xip de la casa Atmel, en comptes de les opcions oferides per altres companyies com Microchip o ARM, va ser la disponibilitat de drivers pel

xip W5100, que ofereix el seu fabricant. Un altre aspecte que va posar a la gama de xips d'Atmel per sobre és la capacitat dels seus microcontroladors de realitzar més operacions per cicle. És important que el programa s'executi tan de pressa com sigui possible, per tal de que la freqüència de missatges DMX enviats es mantingui a un nivell elevat.

Dins dels oferiments de la companyia Atmel, es va optar pel model ATmega 1284p perque es el model més econòmic que incorpora dos canals USART de sortida, necessaris per transmetre dos universos DMX al mateix temps, i també un canal SPI, necessari per comunicar-se amb el xip d'ethernet.

Una altra característica que el fa adequat es l'elevada quantitat de memòria RAM de la que disposa, necessària en el projecte a causa del número de buffers de dades a mantenir dels que fa ús el programa.

Finalment la memòria EEPROM integrada ens permet mantenir les dades de configuració del node, fins i tot després de tallar l'alimentació, sense haver d'afegir un xip integrat addicional al disseny.

4.2. Especificacions del xip d'Ethernet

El xip escollit per dur a terme la tasca de connectar el node a la xarxa d'Ethernet es el W5100, més concretament el mòdul WIZ812MJ, de la companyia Wiznet. Aquest mòdul inclou el circuit bàsic que requereix el xip i el jack RJ45 amb l'interfície necessària.

El xip W5100 es un controlador d'Ethernet compatible amb les configuracions 10BASE-T i 100BASE-T amb capacitat d'autonegociació. El seu hardware incorpora varis protocols de l'stack TCP/IP, com l'UDP i el TCP.

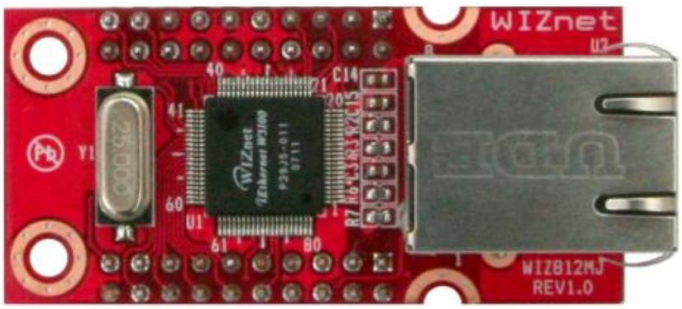

Figura 7. Mòdul d'Ethernet WIZ812MJ

La comunicació amb el microcontrolador del circuit es duu a terme fent ús del bus SPI, que només requereix quatre pins. Els terminals d'entrades i sortides del xip W5100 toleren tensions de 5V, fet pel qual no es necessari adaptar els senyals provinents del microcontrolador.

4.2.1. Justificació de l'elecció

Per poder connectar el microcontrolador a una xarxa Ethernet es van considerar dues opcions. Una era els oferiments de la companyia Wiznet, i l'altre el xip d'Ethernet ENC28J60, de la companyia Microchip. Tot i que estan destinats al mateix fi, difereixen força en les seves capacitats.

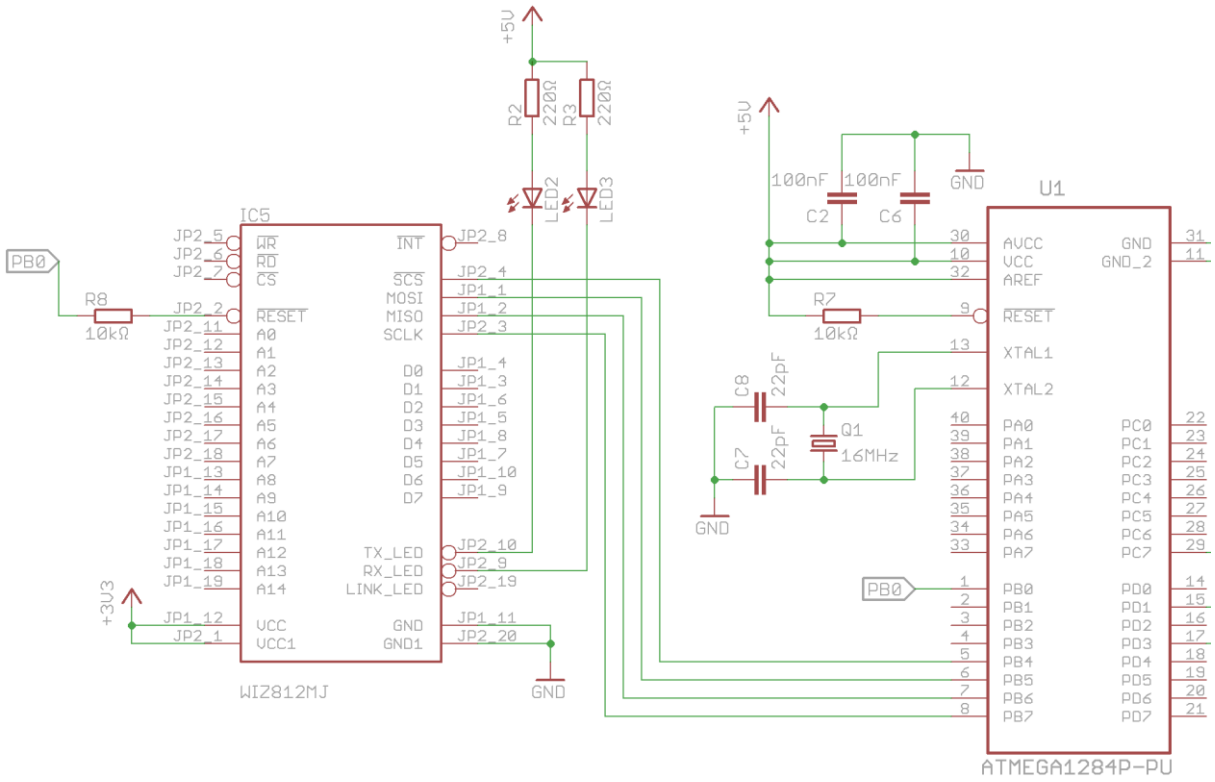

Figura 8. Connexió del xip Ethernet al microcontrolador

El ENC28J60 es un xip que incorpora la capa física i MAC del model OSI. Fent servir un bus SPI, transmet les trames del protocol TCP/IP tal i com arriben, i el microcontrolador ha de realitzar la seva gestió, discernint de quin tipus de paquet es tracta i ha de dur a terme les operacions necessàries per mantenir una connexió, en cas de que sigui un paquet de tipus TCP.

Els xips de la companyia Wiznet (ja sigui el W5100, W5200 o W5300) integren parts del protocol TCP/IP a més de la capa física i d'enllaç. Permeten al microcontrolador accedir a les dades incloses als paquets UDP i TCP sense haver de realitzar les comprovacions d'adreça i CRC, o, en el cas del TCP, establir una connexió de forma manual. Això significa que alleuja la càrrega del microcontrolador, disminuint el número de cicles destintats a comunicar-se amb els altres elements de la xarxa Ethernet i simplifica el firmware del node.

El cost que té la conveniència dels xips de Wiznet es una pèrdua de flexibilitat. Per exemple, el W5100 només permet mantenir quatre sockets simultanis, mentre que una interfície basada en el xip de Microchip es capaç de disposar de tants sockets com sigui necessari, ja que els genera i manté el microcontrolador. En el cas d'aquest projecte aquesta limitació no es relevant, ja que només es fa ús de dos sockets, un de UDP (destinat a la comunicació Art-Net) i un de TCP (destinat al servidor web).

L'elecció del mòdul WIZ812MJ per incorporar el xip W5100 al disseny es deguda a que s'ha cercat simplificar les tasques de prototipatge. En cas de realitzar una producció significativa del node, s'hauria de fer servir el xip W5200, més econòmic i amb millors prestacions que el W5100.

4.3. Especificacions del circuit de sortida de DMX512

El senyal DMX512 es generarà als ports USART del microcontrolador. Les característiques elèctriques d'aquest senyal, però, no són les establertes a l'estàndard DMX512, que fa servir l'estàndard RS-485 a la seva capa física.

L'adaptació dels senyals serà efectuada pel transceptor SN75176. Aquest xip treballa a 5V i el corrent de sortida dels senyals generats pot arribar als 60mA.

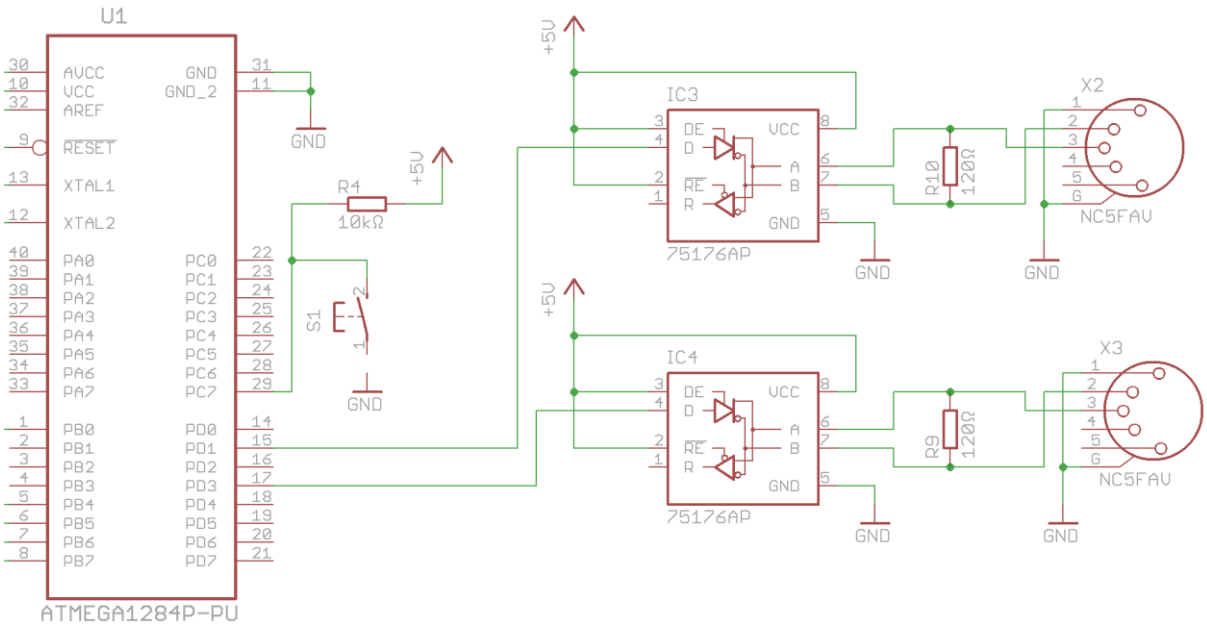

Figura 9. Disseny del circuit de sortida DMX

4.4. Especificacions del circuit d'alimentació

El disseny proposat requereix dos valors diferents de tensió. Per una banda el xip d'Ethernet s'alimenta a 3,3V, i tant el microcontrolador com els transceptors s'alimenten a 5V. Per generar aquestes tensions s'ha fet ús de dos integrats de la casa STMicroelectronics, els reguladors de tensió lineals LD1117V33 i LD1117V50. Tots dos són capaços de generar un ampere de corrent i tenen una tensió de dropout (diferència mínima entre la tensió de sortida i la d'entrada) baixa.

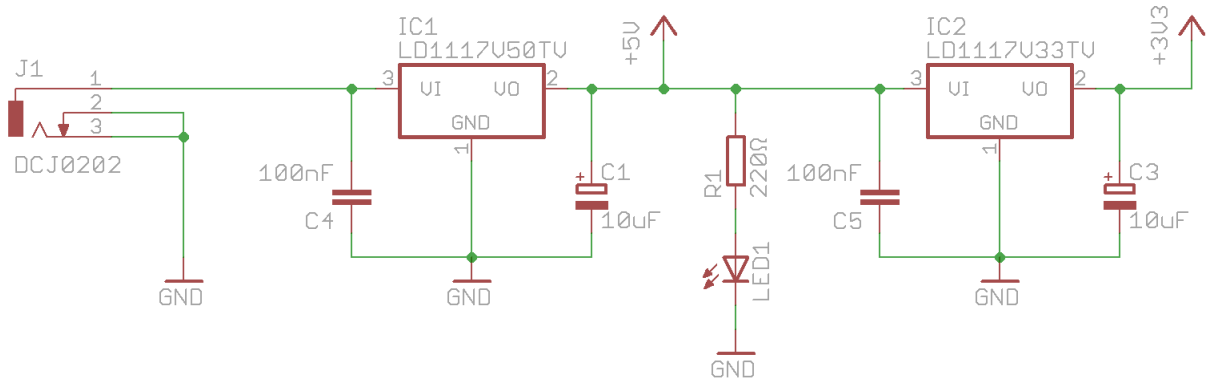

Figura 10. Disseny del circuit d'alimentació

4.4.1. Justificació de l'elecció

A l'hora de regular la tensió d'entrada es presenten dues opcions generals, fer ús d'un regulador de tensió lineal o commutat. A nivell tècnic els reguladors commutats són superiors als lineals, oferint uns nivells d'eficiència molt elevats, però el seu cost és també molt elevat en comparació als lineals.

Un dels objectius del projecte es mantenir un cost per unitat baix, fet pel qual s'optarà pels reguladors de tensió lineal.

Aquesta elecció, però, genera un inconvenient, i es el calor produït pel regulador principal. Una forma d'alleujar aquest problema es alimentar el circuit fins a 9V, però es pretén que el circuit es pugui alimentar a 12V també, així que s'haurà d'instal·lar un dissipador al regulador de 5V per evitar sobreescalfaments. A l'annex C s'inclouen els càlculs de justificació de l'elecció del dissipador.

5. FIRMWARE DEL NODE ART-NET

L'objectiu de la programació del node serà el procés dels missatges que arribin al xip controlador d'Ethernet. S'ha fet ús de la plataforma de desenvolupament integrada Atmel Studio 6 per crear el firmware. Aquesta aplicació disposa d'un compilador de C i C++, capacitat que proporciona una major accessibilitat a l'hora de crear codi complex, al no haver de escriure'l en llenguatge assemblador.

El cicle principal del programa queda representat en el diagrama de flux de la figura 11:

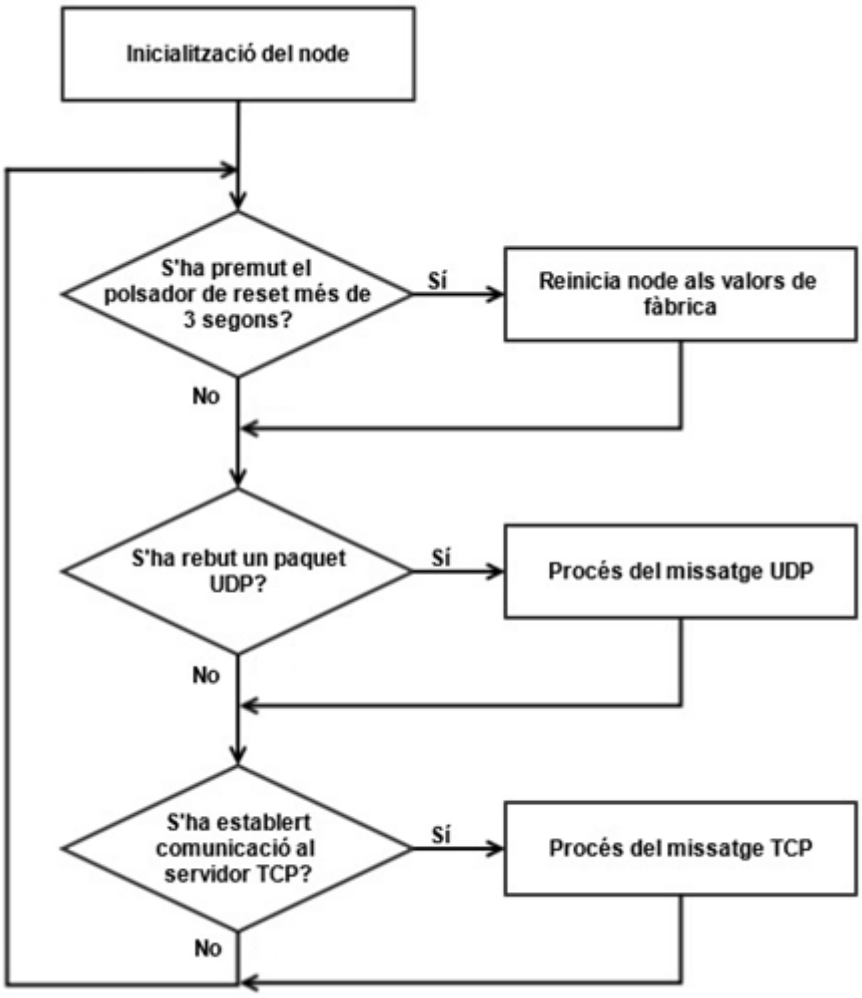

Figura 11. Diagrama de flux general del programa

La inicialització del node consisteix en primer generar les variables necessàries. Després es configuren els ports d'I/O no utilitzats com a sortides a nivell baix, els dos timers de 16 bits,

utilitzats al mode d'unió de trames DMX512, i també els registres relatius als ports USART, utilitzats per transmetre les dades als ports de sortida DMX512.

Finalment s'ha d'inicialitzar el mòdul d'Ethernet. Aquest pas requereix primerament que el terminal de reset del xip W5100 deixi d'estar a nivell alt durant com a mínim 2us. Després cal definir els sockets del xip escrivint en els registres que determinen el protocol al que estaran dedicats i el tamany del seu buffer.

Un cop finalitzades les inicialitzacions ja s'inicia el cicle continu del programa. La primera tasca consisteix en comprovar l'estat del polsador de reset del node i després ja passa a processar els possibles missatges enviats al xip d'Ethernet.

5.1. Procés dels missatges UDP

En cas de que es rebi un paquet UDP de mida superior a 13 bytes, es processaran els seus continguts. El primer pas consisteix en comprovar els primers 8 bytes del paquet tenen els valors de la cadena de caràcters "Art-Net", amb un caràcter nul al final. En cas afirmatiu, s'evalua el codi del tipus de paquet del protocol Art-Net i es compara la mida del paquet rebut. Si la mida coincideix amb el tamany establert pel tipus de paquet, es considera un missatge vàlid i passa a analitzar-se i a efectuar les accions oportunes.

5.1.1. Paquets de tipus Art-DMX

Si el paquet rebut es de tipus Art-DMX, primer s'ha d'avaluar l'adreça del port al que va dirigit. Si els valors de xarxa, subxarxa i univers coincideixen amb els especificats pel node, s'accepta el paquet. El següent pas es desar la IP que ha enviat aquest paquet, necessari per determinar si cal activar el mode d'unió de trames.

El mode d'unio de trames (merge) es activat quan dues adreces IP envien paquets ArtDMX dirigits a un mateix port, entrant les dades de tots dos paquets en conflicte. Hi ha dues formes d'adreçar aquest problema, ignorant les dades de les dues IP o bé unir les dades dels canals DMX512 de totes dues. L'unió es pot efectuar de dues formes, o bé amb el mode d'unió HTP o el mode LTP.

El mode d'unió HTP (Highest Takes Precedence) compara les dades dels canals DMX512 enviats per les IP i afegeix al buffer de transmissió DMX512 el valor més elevat enviat per cada canal. Aquest mode d'unió requereix que es mantinguin dos buffers addicionals, un per cada IP, per tal de dur a terme la comparació.

El mode d'unió LTP (Latest Takes Precedence) envia els canals DMX512 que han estat rebuts més recentment. Aquest mode no ha estat implementat a causa de la impossibilitat de determinar quins canals específics es volen transmetre del paquet ArtDMX. Sense aquesta capacitat, el mode LTP no ofereix res com a mode d'unió de les trames.

Quan es detecta que dues IP han enviat paquets ArtDMX al mateix port de sortida, s'activa el mode d'unió HTP dels canals i es desen les dues adreces IP que han generat el conflicte, de forma que només aquestes poden enviar paquets dirigits al port de sortida on s'ha generat el conflicte.

Al mateix temps s'activa l'interrupció de comparació dels dos timers de 16 bits del microcontrolador. El valor de comparació i el prescaler de tots dos timers s'han configurat de forma que es produeixi una interrupció a cada segon. Aquestes interrupcions es comptabilitzen en una variable, que quan arriba a 10 desactiva el mode merge. Cada cop que una IP diferent a l'anterior envia un paquet ArtDMX al port, la variable es posa a 0. Llavors, quan aquesta arribi a 10 voldrà dir que durant aquests últims 10 segons només una IP s'ha adreçat al port, i per tant el mode d'unió de trames ja no es necessari.

Un cop determinats els canals de la trama DMX512 a enviar, només queda enviar aquestes dades pel port USART del microcontrolador. El senyal de break descrit al capítol dedicat a l'estàndard DMX512 no es pot generar amb el port configurat a la freqüència de transmissió dels canals, 250kbps, i la solució escollida consisteix en reduir aquesta freqüència a 80kbps al iniciar la transmissió, per així complir amb els requeriments de temps. El port USART genera una interrupció cada cop que un byte es enviat, i aquesta es fa servir per actualitzar el buffer de sortida del port amb les dades del següent canal DMX512.

5.1.2. Paquets de tipus ArtPoll

Quan un paquet de tipus ArtPoll es rebut, l'únic camp d'aquest que s'analitza es el de Talk to Me. Aquest camp determina els casos en que el node haurà d'enviar un paquet de tipus ArtPollReply al node controlador de la xarxa. Per defecte aquests paquets només s'han

d'enviar en resposta a paquets de tipus ArtPoll, però el node es pot configurar per enviar paquets ArtPollReply cada cop que un aspecte de la seva configuració canvia.

Després d'avaluar el valor del camp ja descrit, s'envia un paquet de tipus ArtPollReply que conté totes les dades de configuració i descripció del node. Aquest paquet està dirigit al port UDP destinat a Art-Net, 6454, i s'ha d'enviar a l'adreça de broadcast de la xarxa. Per determinar l'adreça de broadcast es fa servir la màscara de subxarxa del node, posant a 1 tots el bits fora de la màscara.

5.1.3. Paquet de tipus ArtAddress

Els paquets de tipus ArtAddress inclouen dades de configuració del node, tals com les adreces dels ports de sortida DMX512. Les dades son extretes del paquet i es passa a reconfigurar el node amb aquestes, desant els valors a la memoria EEPROM per tal de que no es perdin si es talla l'alimentació.

Aquests paquets també es poden fer servir per deshabilitar el mode d'unió de trames. Si aquesta comanda es rebuda, la IP que la va enviar es desa, i només paquets ArtDMX procedents d'aquesta adreça seran acceptats posteriorment.

Finalment, un cop efectuats els canvis al node, s'envia un paquet de tipus ArtPollReply amb les dades del node actualitzades.

5.1.4. Paquets de tipus ArtIpProg

En cas de rebre un paquet de tipus ArtIpProg, s'extreuen les dades de la nova IP o màscara de subxarxa que conté i s'actualitzen els valors de configuració del node, reiniciant el xip controlador d'Ethernet. En resposta a aquest paquet s'ha d'enviar un paquet de tipus ArtIpProgReply. A diferència dels paquets ArtPollReply, aquest s'ha d'enviar a la adreça IP del node controlador, i no a l'adreça de broadcast de la xarxa.

5.2. Procés dels missatges TCP

Quan ja s'han processat els possibles paquets UDP rebuts, es comprova l'estat del socket TCP. Els estats que pot tenir el socket i la relació entre ells queda representada a la figura 12, extreta del full de dades del xip W5100.

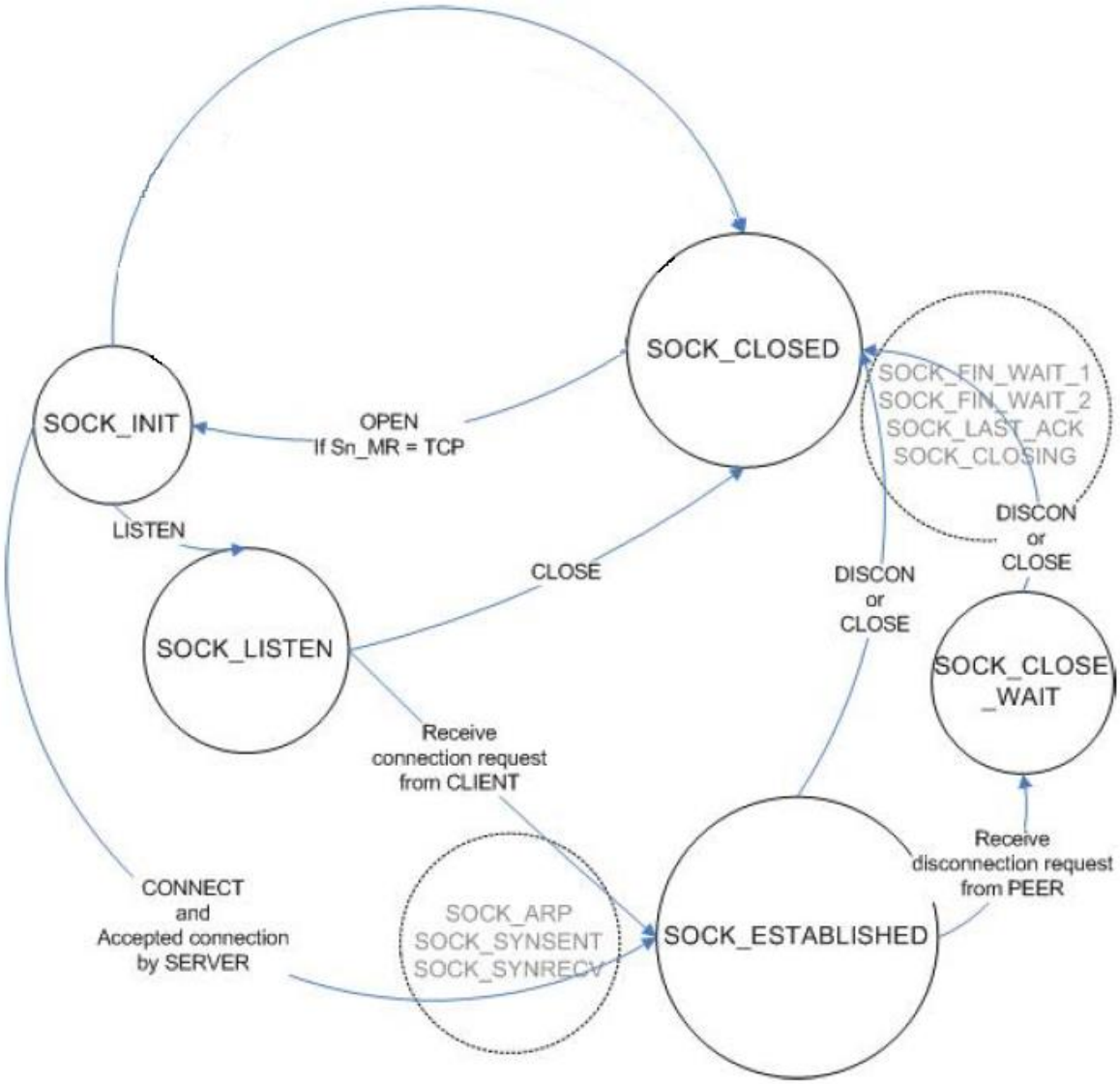

Figura 12. Màquina d'estats del protocol TCP

El socket TCP actua com a servidor, es a dir, espera en estat d'escolta a que un client enviï una sol·licitud de connexió. Si l'estat del socket en el moment de lectura indica que aquest està tancat, el porta a l'estat d'escolta de nou, i en cas de que l'estat sigui de connexió establerta, comprova si ha rebut algun paquet de dades.

En cas d'haver rebut un paquet TCP es cerquen les cadenes de caràcters "GET /" i "POST /". En el primer cas significarà que el missatge es tracta d'una sol·licitud d'un navegador web per obtenir l'informació de la plana web, i per tant es procedirà a la creació d'aquesta i la posterior transmissió.

En el cas de que la cadena "POST /" sigui trobada significarà que a la sol·licitud del navegador a més s'hi troben les dades introduïdes als formularis de la plana web, que el programa farà servir per variar la configuració del node.

Un cop s'hagi acabat d'enviar la plana web al client, el programa passarà a l'estat tancat el socket TCP, per després tornar a deixar-lo en estat d'escolta, on esperarà a rebre altres possibles sol·licituds.

6. INTERFICIE D'USUARI

Els usuaris del node Art-Net disposen de tres vies per canviar la configuració del node. La primera consisteix en l'ús de missatges Art-Net de configuració. D'aquesta forma, mitjançant l'ús de missatges de tipus ArtAddress y ArtIpProg es poden canviar la IP del node, la seva màscara de subxarxa i els universos que es transmeten.

El principal desavantatge d'aquest primer mètode es troba en el fet de que no tot el programari destinat a transmetre trames Art-Net incorpora tot el protocol. A vegades només són capaços de transmetre missatges de tipus ArtDMX. Per aquest motiu es va decidir incorporar un servidor web al firmware del node. D'aquesta forma fent ús de qualsevol navegador web es pot realitzar la configuració de tots els paràmetres més importants del node.

La plana web generada permet visualitzar els valors actuals dels paràmetres per una banda, i per una altra els valors que l'usuari pretén introduir al node. Aquestes dades s'actualitzen quan l'usuari envia les dades dels formularis al servidor.

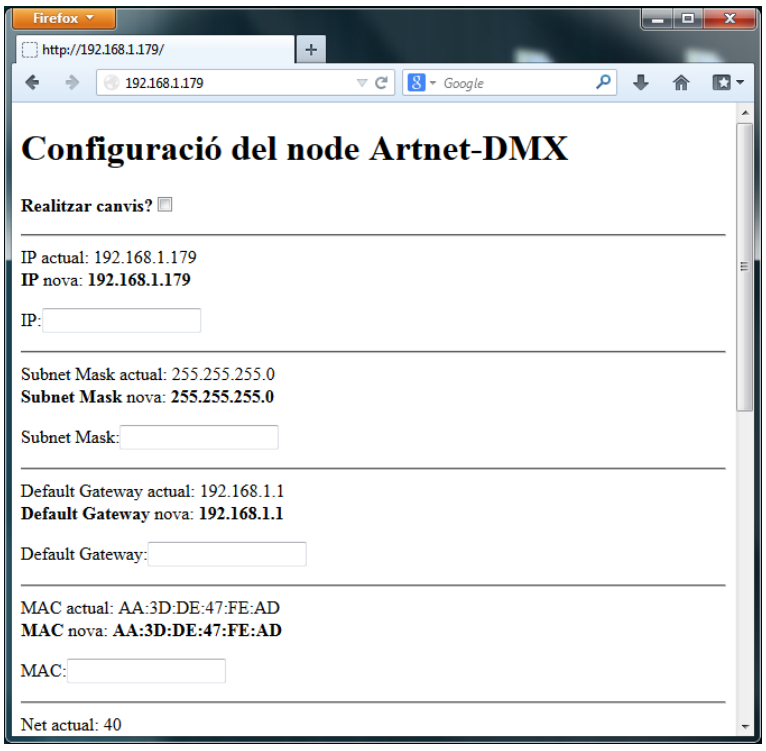

Figura 13. Plana web de configuració del node

Un cop s'estigui satisfet amb els canvis que s'efectuaran, s'ha de seleccionar la casella de verificació situada a la part superior de la plana web i tornar a enviar els formularis al servidor (no fa falta omplir-los de nou). Després d'efectuar aquest pas, el controlador Ethernet reinicia els sockets, així que s'ha de tornar a carregar la plana web manualment. Un cop torni a carregar es veurà com els valors marcats com a actuals han variat en cas de modificació de l'usuari.

Finalment existeix una tercera forma de canviar la configuració del node. A l'exterior del node es pot trobar un polsador. Si aquest es prem durant més de tres segons, la configuració del node tornarà a uns valors de fàbrica especificats al firmware.

Aquesta funcionalitat té com a objecte desfer canvis realitzats mitjançant qualsevol dels dos altres mètodes, en cas de que aquests impossibilitin l'acces al node pels mitjans habituals. Per exemple si es canvia la IP a una altra fora del rang establert pel router per a la xarxa local. En aquest cas també es pot canviar la configuració del router per poder tenir accès de nou al node i així desfer els canvis, però acostuma a ser una solució més laboriosa.

Els valors de fàbrica especificats al firmware queden definits a la següent taula:

Paràmetre	**Valor**
IP	2.0.0.100
Màscara de subxarxa	2.0.0.0
Gateway per defecte	2.0.0.1
MAC	AA:3D:DE:47:FE:AD
Xarxa	0
Subxarxa	0
Univers Port 0	0
Univers Port 1	0
Nom curt	"Node 1 ArtNet-DMX"

Taula 5. Valors de fàbrica del node

7. COMPROVACIONS DEL FUNCIONAMENT

A l'hora de testar el funcionament del node s'han fet servir dos programes principalment. El primer és DMX-Workshop. Aquesta aplicació, disponible de forma gratuïta a la plana web d'Artistic Licence, ens permet realitzar un diagnòstic de la xarxa Art-Net i també dur a terme la configuració dels nodes que la composen.

L'altre aplicació utilitzada és Wireshark, programa fet servir per analitzar el tràfic d'una xarxa de comunicacions, identificant els tipus de paquets enviats i rebuts i els camps que inclouen. Wireshark incorpora a les seves definicions de paquets el protocol Art-Net, fet que simplifica l'anàlisi de paquets rebuts i enviats pel node. En cas de que qualsevol camp dels missatges elaborats pel node no sigui correcte, es pot detectar l'error i corregir-lo amb facilitat.

A l'hora d'escriure el firmware, la primera comprovació que es va dur a terme es la d'identificació del node a la xarxa. Que el node aparegui a la llista d'elements connectats del programa significa que el node ha processat de forma adequada el missatge ArtPoll i ha contestat amb ArtPollReply.

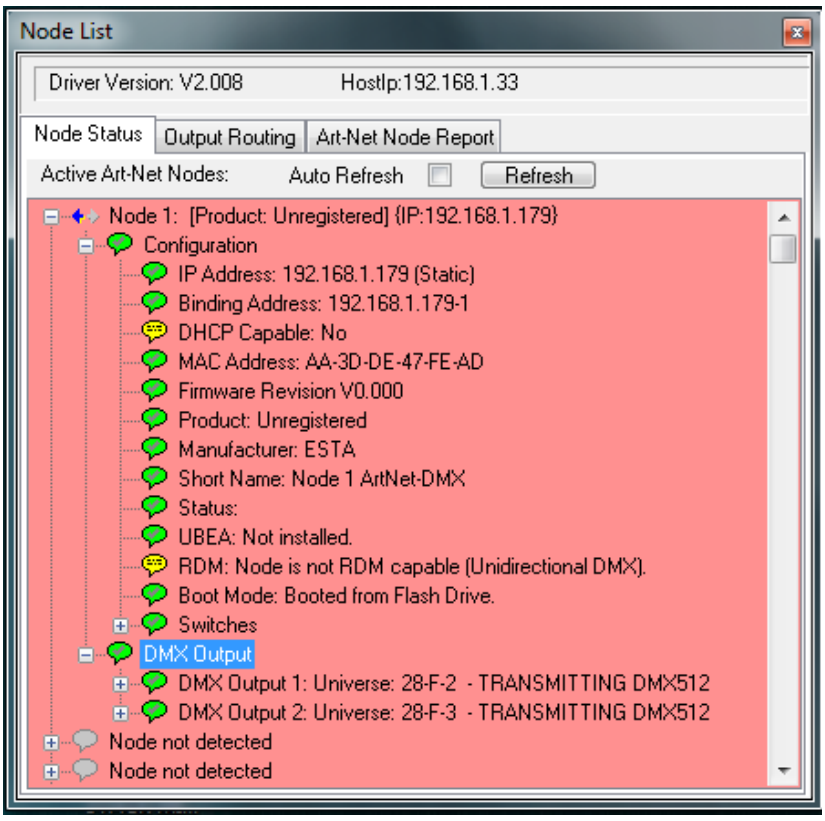

Figura 14. Detecció del node Art-Net a la xarxa

Monitoritzant la xarxa amb Wireshark podem observar el temps de resposta del node i el contingut del paquet enviat.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	192.168.1.35	192.168.1.255	ARTNET	56	ArtPoll (0x2000)
2	0.04154500	192.168.1.177	192.168.1.255	ARTNET	282	ArtPollReply (0x2100)

Figura 15. Monitorització del temps de resposta del node a un paquet ArtPoll

La següent comprovació realitzada va consistir en verificar tant que el node rebia els paquets ArtDMX de forma adequada i generava el senyal DMX512 als seus canals de sortida amb les dades enviades al paquet. El programa DMX-Workshop disposa de dues utilitats per enviar paquets ArtDMX, ja sigui a una direcció IP determinada (que pot ser de broadcast) o als nodes identificats de la xarxa (transmissió unicast). La primera utilitat permet manipular manualment el contingut de cada canal (Transmit Preset), i la segona genera automàticament i de forma dinàmica el contingut dels canals a enviar (Transmit Dynamic).

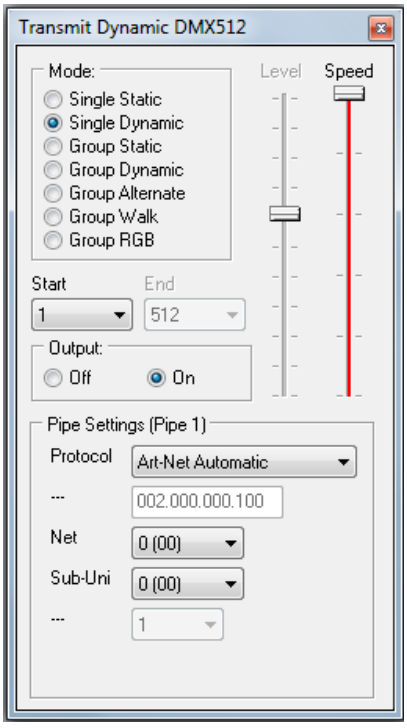

Figura 16. Utilitat del DMXWorkshop per enviar trames de forma dinàmica

En cas de que es faci ús del tipus de transmissió Art-Net automàtica, s'haurà de seleccionar l'univers (incloent la xarxa i subxarxa d'aquest) assignat al node per tal de que els missatges siguin enviats a la seva IP.

Per comprovar que la transmissió de dades DMX512 es realitzava d'acord a les especificacions del protocol s'ha fet ús d'un dispositiu anomenat Tester DMX512. Aquest dispositiu és el producte del projecte de final de carrera d'un ex-alumne de la universitat, Josep Maria Busquets. El tester ens permet visualitzar els valors dels canals de les trames DMX512 en temps real. Cada cop que es desplacin els sliders del DMX-Workshop, s'hauria de visualitzar un canvi en els valors dels canals. D'aquesta forma es va comprovar el correcte funcionament de tots dos canals de sortida del node.

Després es va dur a terme la comprovació de que els mitjans de reconfiguració del node eren efectius. Realitzant canvis mitjançant la plana web de configuració, o missatges de tipus ArtAddress o ArtIpProg, aquests s'haurien de visualitzar als paràmetres del node que aquest fa conèixer amb els missatges de tipus ArtPollReply.

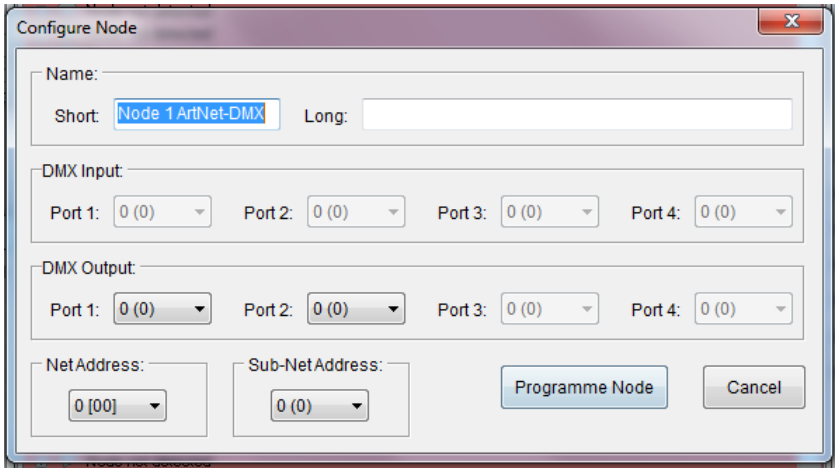

Figura 17. Utilitat per enviar paquets ArtAddress del DMX-Workshop

Finalment es va creure oportú realitzar una prova de rendiment del node que va tindre per objecte verificar a quina freqüència eren transmeses les trames DMX512. Es temia que el temps de procés dels missatges Art-Net produis un retard en aquest aspecte.

El màxim teòric de transmissió de trames del protocol DMX es de 61,035 trames per segon. Existeix la possibilitat de que un espectacle luminotècnic contingui efectes que requereixen

una freqüència de refresc elevada per tal de ésser efectius, així que pot arribar a ser un paràmetre crític.

El mètode ideat per mesurar la freqüència de transmissió va consistir en modificar el firmware per tal de que cada cop que deu trames DMX fossin transmeses a un port de sortida, el node enviara un paquet UDP sense cap contingut en resposta. Wireshark permet visualitzar el temps en que un paquet ha estat transmès o enviat, i s'ha fet servir aquesta informació per obtenir un valor aproximat de la freqüència. Els paquets ArtDMX s'enviaran a una freqüència de 40 trames per segon, que es el màxim que permet la utilitat d'enviar paquets DMX de forma dinàmica del DMX-Workshop.

Es van realitzar mesures per tres casos diferents de funcionament del node. El primer cas consisteix en que tots dos ports de sortida del node transmeten el mateix univers. Les mesures van comprovar que tots els missatges enviats al node eren transmesos, així que la freqüència de transmissió de trames DMX pels ports de sortida era de 40 trames per segon.

No.	Time	Source	Destination	Protocol	Length	Info
27	0.57894600	192.168.1.35	192.168.1.177	ARTNET	572	ArtDMX (0x5000)
28	0.58958900	192.168.1.177	192.168.1.35	UDP	60	Source port: 6454 Destination port: 6454
29	0.60396000	192.168.1.35	192.168.1.177	ARTNET	572	ArtDMX (0x5000)
30	0.62809100	192.168.1.35	192.168.1.177	ARTNET	572	ArtDMX (0x5000)
31	0.62913100	192.168.1.35	192.168.1.177	ARTNET	572	ArtDMX (0x5000)
32	0.65493500	192.168.1.35	192.168.1.177	ARTNET	572	ArtDMX (0x5000)
33	0.67997000	192.168.1.35	192.168.1.177	ARTNET	572	ArtDMX (0x5000)
34	0.70575700	192.168.1.35	192.168.1.177	ARTNET	572	ArtDMX (0x5000)
35	0.73095700	192.168.1.35	192.168.1.177	ARTNET	572	ArtDMX (0x5000)
36	0.75594800	192.168.1.35	192.168.1.177	ARTNET	572	ArtDMX (0x5000)
37	0.78095500	192.168.1.35	192.168.1.177	ARTNET	572	ArtDMX (0x5000)
38	0.80598800	192.168.1.35	192.168.1.177	ARTNET	572	ArtDMX (0x5000)
39	0.83094900	192.168.1.35	192.168.1.177	ARTNET	572	ArtDMX (0x5000)
40	0.84138400	192.168.1.177	192.168.1.35	UDP	60	Source port: 6454 Destination port: 6454
41	0.85596200	192.168.1.35	192.168.1.177	ARTNET	572	ArtDMX (0x5000)

Figura 18. Visualització de la resposta del node al transmetre 10 trames DMX512

El segon cas va consistir en la transmissió de dos universos diferents a cada port de sortida del node. En aquest cas la freqüència de transmissió es va veure reduïda al node a una mitjana aproximada de 32 trames per segon. Aquesta reducció es deguda al coll d'ampolla que suposa la transmissió de dades pel bus SPI del xip d'Ethernet, la freqüència màxima del qual es 300.000kbps. Tot i això es una freqüència prou elevada per la majoria d'efectes luminotècnics.

Finalment es va voler provar el rendiment del mode merge (unió) de trames. Aquest cas va originar els pitjors resultats, amb una freqüència de trasmissió mitjana de 7 trames per

segon, valor que descarta aquesta funcionalitat en cas de necessitar un temps de refresc baix. El baix valor d'aquest mode es degut principalment a la velocitat de processament del microcontrolador. El mode merge exigeix comparar continuament buffers de dades, tasca que requereix un número important de cicles del processador. En cas de voler fer ús extensiu d'aquesta funcionalitat, s'hauria de considerar un microcontrolador més ràpid.

8. RESUM DEL PRESSUPOST

L'import total del projecte és de nou-cents cinquanta-cinc euros amb seixanta-sis cèntims, sense IVA.

9. CONCLUSIONS

Es considera que els objectius marcats inicialment han estat complerts. S'ha dut a terme el disseny del node Art-Net, a nivell de hardware i software. Les capacitats d'aquest inclouen la possibilitat de transmetre dos universos DMX512 de sortida i l'implementació necessària del protocol per tal de poder integrar-se en una xarxa Art-Net amb la versió més actual del protocol.

S'han implementat també varis mètodes de configuració del node, que pretenen fer més fàcil l'interacció d'aquest amb l'usuari.

De cares a l'elecció dels components, s'ha tingut sempre en consideració el cost per unitat del node. Es creu que el disseny compleix amb aquest objectiu, tenint un cost per unitat que, sense tenir en compte descomptes de quantitat, és inferior a les ofertes de dispositius similars d'altres companyies.

El disseny proposat pot ser objecte de varies futures millores. En cas de voler implementar funcionalitats RDM, aquestes es podrien obtenir fent modificacions molt menors al hardware (entrada XLR) i afegint el procés dels nous tipus de paquets al firmware.

L'altre camp de millora del node es troba en la manufactura del disseny. Fer ús de components de muntatge superficial o l'integracio del xip d'Ethernet sense necessitat d'un mòdul prefabricat (opcions descartades per fer més simple el prototipatge) ajudarien a reduir tant el cost per unitat com el tamany del node.

Xavier Herranz Izquierdo
Graduat en Enginyeria Electrònica Industrial i Automàtica

Figueres, 16 de gener de 2014

10. RELACIÓ DE DOCUMENTS

Els documents que conformen el projecte són la memòria, els plànols, el plec de condicions, l'estat d'amidaments i el pressupost.

11. BIBLIOGRAFIA

AAVID THERMALLOY. ML7G Datasheet (http://docs-europe.electrocomponents.com/webdocs/0e78/0900766b80e78bcf.pdf, 15 d' octubre de 2013)

ARTISTIC LICENCE. Specification for the Art-Net 3 Ethernet Communication Standard (http://www.artisticlicence.com/WebSiteMaster/User%20Guides/art-net.pdf, 15 d' octubre de 2013)

ATMEL. ATmega1284P Datasheet (http://docs-europe.electrocomponents.com/webdocs/0dc5/0900766b80dc5089.pdf, 15 d' octubre de 2013)

STEVENS, W.R. TCP/IP Illustrated, Vol. 1: The Protocols. 2a ed. Addison-Wesley Professional, 1994.

STMICROELECTRONICS. LD1117xx Datasheet (http://docs-europe.electrocomponents.com/webdocs/0dbd/0900766b80dbda35.pdf, 15 d' octubre de 2013)

TEXAS INSTRUMENTS. SN75176A Datasheet (http://www.farnell.com/datasheets/1700985.pdf, 15 d' octubre de 2013)

WIZNET INC.. WIZ812MJ Datasheet (http://docs-europe.electrocomponents.com/webdocs/0dc7/0900766b80dc7d24.pdf, 15 d' octubre de 2013)

WIZNET INC.. W5100 Datasheet (https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf, 15 d' octubre de 2013)

12. GLOSSARI

Broadcast: Forma de transmissió de dades on el node emissor fa arribar l'informació a tots els receptors de forma simultanea.

CRC: Cyclic Redundancy Check. Codi de detecció de errors.

DHCP: Dynamic Host Configuration Protocol. Protocol de xarxa que permet als nodes d'aquesta obtenir els seus paràmetres de configuració de forma automàtica.

EEPROM: Electrically Erasable Programmable Read-Only Memory. Tipus de memoria no volàtil.

Socket: Concepte abstracte pel qual dos elements d'una xarxa poden intercanviar dades. Es composa d'una adreça IP, un protocol i un port.

SPI: Serial Peripheral Interface. Estàndard de comunicacions, que es duen a terme en sèrie i de forma síncrona.

TCP: Transmision Control Protocol. Protocol del nivell de transport del model OSI. És orientat a la connexió, fet que qualifica el protocol com a fiable.

UDP: User Datagram Protocol. Protocol del nivell de transport del model OSI. No disposa de mecanismes de control de l'entrega de missatges, fet que el qualifica com a no fiable.

Unicast: Forma de transmissió de dades on el node emissor envia l'informació a una única destinació.

USART: Universal Synchronous/Asynchronous Receiver/Transmitter. Peça de maquinari que tradueix dades entre formes paral·leles i serials, ja sigui de forma síncrona o asíncrona.

VPN: Virtual Private Network. Tecnologia de xarxa que permet extendre una xarxa local (LAN) sobre una xarxa pública o no controlada com Internet.

A. CODI DEL PROGRAMA

```
/*Node ArtNet-DMX amb capacitat per a transmetre dos universos DMX i servidor web
per dur a terme la configuració*/

#include <capcalera.h>

/*Inicialització de les variables no volàtils a la memoria EEPROM*/

uint8_t EEMEM ip_node_eeprom[4] = {2,0,0,100};
uint8_t EEMEM subnet_mask_eeprom[4] = {255,0,0,0};
uint8_t EEMEM gateway_node_eeprom[4] = {2,0,0,1};
uint8_t EEMEM mac_node_eeprom[6] = {0xAA, 0x3D, 0xDE, 0x47, 0xFE, 0xAD};
uint8_t EEMEM net_eeprom = 0;
uint8_t EEMEM subnet_eeprom = 0;
uint8_t EEMEM univers0_eeprom = 0;
uint8_t EEMEM univers1_eeprom = 0;
char EEMEM nom_eeprom[18] ="Node 1 ArtNet-DMX";

/*Valors de fàbrica del node. En cas de reinici es tornaria a aquests valors*/

const uint8_t ip_node_fab[4] = {2,0,0,100};
const uint8_t subnet_mask_node_fab[4] = {255,0,0,0};
const uint8_t gateway_node_fab[4] = {2,0,0,1};
const uint8_t mac_node_fab[6] = {0xAA, 0x3D, 0xDE, 0x47, 0xFE, 0xAD};
const uint8_t net_fab = 0;
const uint8_t subnet_fab = 0;
const uint8_t univers0_fab = 0;
const uint8_t univers1_fab = 0;
const char nom_fab[18] = {"Node 1 ArtNet-DMX"};

/*Variables de configuació de la connexió i del node*/

unsigned char mac_node[6] = {0xAA, 0x3D, 0xDE, 0x47, 0xFE, 0xAD};
unsigned char ip_node[4] = {2,0,0,100};
unsigned char subnet_mask_node[4]= {255, 0, 0, 0};
unsigned char gateway_node[] = {2,0,0,1};
unsigned char net=0;
unsigned char subnet=0;
unsigned char univers0=0;
unsigned char univers1=0;
char nom[18]={"Node 1 ArtNet-DMX"};

/*Variables de configuració del node. Mostren el valor a la web que prendrà la
variable un cop s'efectuin els canvis*/

uint16_t ip_node_web[4];
uint16_t subnet_mask_node_web[4];
uint16_t gateway_node_web[4];
uint16_t mac_node_web[6];
uint16_t net_web;
uint16_t subnet_web;
uint16_t univers0_web;
uint16_t univers1_web;
char nom_web[18];

/*IP font dels missatges UDP i TCP rebuts. Els missatges de resposta s'enviaran a
aquesta mateixa direcció*/

uint8_t ip_remota[4];
uint8_t ip_remota_broad[4];

/*Variables de recepció i enviament de paquets UDP i TCP*/

uint16_t restant;
uint16_t offset;
```

```
int mida_paquet_udp;
uint8_t buffer_udp [530];
uint16_t mida_paquet_tcp=0;
uint8_t estat_tcp;
uint8_t buffer_tcp[2000];
uint16_t tipus_paquet;

/*Variable de comptatge de temps de reset*/
uint8_t compta=0;

/*Variable que determina quan enviar missatges de tipus ArtPollReply*/
uint8_t talktome=0;

/*Cadena que marca l'inici de tots el paquets del protocol Art-Net*/
const uint8_t artnet_inici[8]={'A','r','t','-','N','e','t',0x00};

char conversio_str[15];
char * index;
int get_present,post_present;

/*Variables relatives a la funció d'unir (merge) paquets ArtDMX*/
volatile uint8_t estat_merge_0=0;
volatile uint8_t estat_merge_1=0;
uint8_t ip_remota_merge_00[4];
uint8_t ip_remota_merge_01[4];
uint8_t ip_remota_merge_10[4];
uint8_t ip_remota_merge_11[4];
volatile uint8_t identificacio_ip_00=0;
volatile uint8_t identificacio_ip_01=0;
uint8_t identificacio_ip_10=0;
uint8_t identificacio_ip_11=0;
uint8_t buffer_udp_merge_00[512];
uint8_t buffer_udp_merge_01[512];
uint8_t buffer_udp_merge_10[512];
uint8_t buffer_udp_merge_11[512];
volatile uint8_t segons_inactivitat_0=0;
volatile uint8_t segons_inactivitat_1=0;
volatile uint8_t torn_inactivitat_0=0;
volatile uint8_t torn_inactivitat_1=0;
uint8_t ip_remota_cancelacio[4];
uint8_t cancela_merge=0;
/*Variables relatives a la transmissió de DMX via el port USART*/

volatile int16_t ptr_dmx_0 = 0;
volatile int16_t ptr_dmx_1= 0;
volatile uint8_t estat_dmx_0 = 0;
volatile uint8_t estat_dmx_1= 0;
uint8_t buffer_dmx0[buffer_max];
uint8_t buffer_dmx1[buffer_max];
uint8_t dmx_llest_0=0;
uint8_t dmx_llest_1=0;

/*Variables de les estructures dels missatges del protocol Art-Net, fetes servir
per a la transmissió i interpretació d'aquests*/

paquet_pollreply_t paquet_pollreply;
paquet_pollreply_t *paquet_pollreply_p;

paquet_artaddress_t paquet_artaddress;
paquet_artaddress_t *paquet_artaddress_p;

paquet_artipprog_t paquet_artipprog;
paquet_artipprog_t *paquet_artipprog_p;

paquet_ipprogreply_t paquet_ipprogreply;
paquet_ipprogreply_t *paquet_ipprogreply_p;

int conta_prova=0;
```

```
int conta_prova1=0;

int main(void)
{

_delay_ms(100);
/*Els pins no fets servir es configuren com a sortides a nivell baix*/
DDRA=0xFF;
DDRB=0xFF;
DDRC=0x7F;
DDRD=0XFF;
PORTA=0x00;
PORTB=0x00;
PORTC=0x00;
PORTD=0x00;
/*Seqüència de reset necessaria per inicialitzar el mòdul d'Ethernet*/
PORTB=1;
_delay_ms(1);
PORTB=0;
_delay_ms(1);
PORTB=1;

/*Inicialització del xip d'ethernet i els ports USART*/
iniciar_dades_web();
iniciar_sockets();
iniciar_usart();
/*S'habiliten les interrupcions globals*/
sei();

/*Configuració de les interrupcions dels timers*/
TCCR1B|=(1<<WGM12);
OCR1A=15624;
TCCR1B|=((1<<CS12)|(1<<CS10));
TCCR3B|=(1<<WGM32);
OCR3A=15624;
TCCR3B|=((1<<CS32)|(1<<CS30));

/*Inicialització de buffers*/
memset(buffer_udp_merge_00,0,512);
memset(buffer_udp_merge_01,0,512);
memset(buffer_udp_merge_10,0,512);
memset(buffer_udp_merge_11,0,512);

for(;;)
{
/*Si s'ha premut el polsador de reset, es retorna el node als valors de fàbrica*/
/*Si s'ha premut durant aproximadament 3 segons el polsador de reset, es retorna el
node als valors de fàbrica*/
if((PINC&(1<<PINC7))==0)
{
while((PINC&(1<<PINC7))==0)
{
_delay_ms(500);
compta++;
}
if (compta>6)
{
reinici_valors();
}
while((PINC&(1<<PINC7))==0)
{

}
/*Es deixa passar un quart de segon per evitar el rebot del polsador*/
_delay_ms(250);
compta=0;
}
```

```
/*Es comprova si s'ha rebut un paquet UDP i si es de més de 14 bytes (missatge
d'Art-net més petit) es passa a analitzar*/
mida_paquet_udp = extreure_udp();
if(mida_paquet_udp>13)
{
/*Es comprova el tipus de missatge Art-Net i es procedeix en funció d'aquest*/
tipus_paquet= comprovar_tipus();
/*En cas de que sigui un missatge de tipus ArtDMX es comprova si el port és un dels
dos configurats al node i en cas afirmatiu es transmet la trama DMX via el port
USART*/
if ((tipus_paquet==codi_artdmx) && (mida_paquet_udp==mida_artdmx))
{
/*Si el paquet està dirigit al univers del port 0*/
if(((buffer_udp[14]==(univers0+(subnet<<4))) && buffer_udp[15]==net))
{
if(cancela_merge==0 || memcmp(ip_remota,ip_remota_cancelacio,4)==0)
{
if(dmx_llest_0==0)
{

if(identificacio_ip_00==0)
{
memcpy(ip_remota_merge_00,ip_remota,4);
identificacio_ip_00=1;
}
if((memcmp(ip_remota,ip_remota_merge_00,4)!=0) & (identificacio_ip_00==1) &
(identificacio_ip_01==0))
{
estat_merge_0=estat_merge_0+0x08;
memcpy(ip_remota_merge_01,ip_remota,4);
identificacio_ip_01=1;
}

if(estat_merge_0>=0x08)
{
/*S'habilita l'interrupció de comparació del timer 1*/
TIMSK1|=(1<<OCIE1A);
/*mode HTP de merge*/
if((estat_merge_0 & 0x02)==0)
{
if(memcmp(ip_remota,ip_remota_merge_00,4)==0)
{
if(torn_inactivitat_0==1)
{
segons_inactivitat_0=0;
torn_inactivitat_0=0;
}
for(int i=0;i<512;i++)
{
if(buffer_udp[i+18]>=buffer_udp_merge_01[i])
{
buffer_dmx0[i]=buffer_udp[i+18];
}
memcpy(buffer_udp_merge_00,buffer_udp+18,512);
}
}
if(memcmp(ip_remota,ip_remota_merge_01,4)==0)
{
if(torn_inactivitat_0==0)
{
segons_inactivitat_0=0;
torn_inactivitat_0=1;
}
for(int i=0;i<512;i++)
{
if(buffer_udp[i+18]>=buffer_udp_merge_00[i])
{
buffer_dmx0[i]=buffer_udp[i+18];
```

```
}
memcpy(buffer_udp_merge_01,buffer_udp+18,512);
}
}

}else
{
/*Mode de merge LTP no implementat*/
}
/*si es compten 10 segons seguits, es desactiva el mode merge i l'interrupció*/
if(segons_inactivitat_0>=10)
{

estat_merge_0=0;
identificacio_ip_00=0;
identificacio_ip_01=0;
TIMSK1&=~(1<<OCIE1A);
torn_inactivitat_0=0;
segons_inactivitat_0=0;
}
}else
{
memcpy(buffer_dmx0, &buffer_udp[18],512);
}
dmx_llest_0=1;
}

if(estat_dmx_0==0)
{
conta_prova++;
if(conta_prova==10)
{
inici_paquet_udp(ip_remota, port_artnet);
escriure_udp(0);
terminar_paquet_udp();
conta_prova=0;
}

dmx_llest_0=0;
envia_dmx_0();
}
}
}

/*Si el paquet ArtDMX està dirigit a l'univers el port 1*/
if((buffer_udp[14]==(univers1+(subnet<<4))) && buffer_udp[15]==net)
{
if(cancela_merge==0 || memcmp(ip_remota,ip_remota_cancelacio,4)==0)
{
if(dmx_llest_1==0)
{

if(identificacio_ip_10==0)
{
memcpy(ip_remota_merge_10,ip_remota,4);
identificacio_ip_10=1;
}
if((memcmp(ip_remota,ip_remota_merge_10,4)!=0) & (identificacio_ip_10==1) &
(identificacio_ip_11==0))
{
estat_merge_1=estat_merge_1+0x08;
memcpy(ip_remota_merge_11,ip_remota,4);
identificacio_ip_11=1;
}

if(estat_merge_1>=0x08)
{
TIMSK3|=(1<<OCIE3A);
```

```
if((estat_merge_1 & 0x02)==0)
{
if(memcmp(ip_remota,ip_remota_merge_10,4)==0)
{
if(torn_inactivitat_1==1)
{
segons_inactivitat_1=0;
torn_inactivitat_1=0;
}
for(int i=0;i<512;i++)
{
if(buffer_udp[i+18]>=buffer_udp_merge_11[i])
{
buffer_dmx1[i]=buffer_udp[i+18];
}
memcpy(buffer_udp_merge_10,buffer_udp+18,512);
}
}
if(memcmp(ip_remota,ip_remota_merge_11,4)==0)
{
if(torn_inactivitat_1==0)
{
segons_inactivitat_1=0;
torn_inactivitat_1=1;
}
for(int i=0;i<512;i++)
{
if(buffer_udp[i+18]>=buffer_udp_merge_10[i])
{
buffer_dmx1[i]=buffer_udp[i+18];
}
memcpy(buffer_udp_merge_11,buffer_udp+18,512);
}
}

}else
{
/*Mode merge LTP no implementat*/
}
/*si es compten 10 segons seguits, es desactiva el mode merge i l'interrupció*/
if(segons_inactivitat_1>=10)
{
estat_merge_1=0;
identificacio_ip_10=0;
identificacio_ip_11=0;
TIMSK3&=~(1<<OCIE3A);
torn_inactivitat_1=0;
segons_inactivitat_1=0;
}
}else
{
memcpy(buffer_dmx1, &buffer_udp[18],512);
}
dmx_llest_1=1;
}

if(estat_dmx_1==0)
{
dmx_llest_1=0;
envia_dmx_1();
}
}
}
}

/*En cas d'esser un missate de tipus ArtPoll s'enviarà un missatge de resposta
ArtPollReply*/
if ((tipus_paquet==codi_poll) && (mida_paquet_udp==mida_poll))
```

```
{
talktome=(buffer_udp[12]&(0x02));
enviar_pollreply();
}
/*Si es un missatge de tipus ArtAddress s'efectuaran els canvis en el node en
funció del contingut del missatge*/
if ((tipus_paquet==codi_address) && (mida_paquet_udp==mida_address))
{
paquet_artaddress_p=&paquet_artaddress;
memcpy(paquet_artaddress_p,buffer_udp,mida_address);
if(paquet_artaddress_p->netswitch & 0x80)
{
net_web=(paquet_artaddress_p->netswitch & 0x7F);
}
if(paquet_artaddress_p->subswitch & 0x80)
{
subnet_web=(paquet_artaddress_p->subswitch & 0x7F);
}
if(paquet_artaddress_p->swout[0] & 0x80)
{
univers0_web=(paquet_artaddress_p->swout[0] & 0x7F);
}
if(paquet_artaddress_p->swout[1] & 0x80)
{
univers1_web=(paquet_artaddress_p->swout[1] & 0x7F);
}
/*En cas de rebre la comanda de parar el mode merge, es desa l'adreça d'on prové el
missatge per acceptar ArtDMX
només d'aquesta, i també es desactiven les interrupcions*/
if(paquet_artaddress_p->command & 0x01)
{
TIMSK1&=~(1<<OCIE1A);
TIMSK3&=~(1<<OCIE3A);
cancela_merge=1;
memcpy(ip_remota_cancelacio,ip_remota,4);
estat_merge_0=0;
estat_merge_1=0;

}
memcpy(nom_web,paquet_artaddress_p->shortname,18);
/*un cop desats els valors del missatge, s'actualitza l'EEPROM i s'envia un
missatge de tipus ArtPollReply de resposta*/
actualitzar_eeprom(0);
enviar_pollreply();

}
/*Si es un missatge de tipus ArtIpProg s'efectuaran els canvis en el node en funció
del contingut del missatge*/
if ((tipus_paquet==codi_ipprog) && (mida_paquet_udp==mida_ipprog))
{

paquet_artipprog_p=&paquet_artipprog;
memcpy(paquet_artipprog_p,buffer_udp,mida_ipprog);
if((paquet_artipprog_p->command & 0x82)==0x82)
{
subnet_mask_node[3]=paquet_artipprog_p->progsm0;
subnet_mask_node[2]=paquet_artipprog_p->progsm1;
subnet_mask_node[1]=paquet_artipprog_p->progsm2;
subnet_mask_node[0]=paquet_artipprog_p->progsm3;
}
if((paquet_artipprog_p->command & 0x84)==0x84)
{
ip_node[3]=paquet_artipprog_p->progip0;
ip_node[2]=paquet_artipprog_p->progip1;
ip_node[1]=paquet_artipprog_p->progip2;
ip_node[0]=paquet_artipprog_p->progip3;
}
```

```
/*Si l'objectiu del missatge era canviar l'IP o la màscara de subxarxa,
s'actualitzarà l'EEPROM amb aquests valors*/
if((paquet_artipprog_p->command & 0x82)==0x82 || (paquet_artipprog_p->command &
0x84)==0x84)
{
actualitzar_eeprom(1);
}
/*Si l'objectiu del missatge era tornar als valors de fàbrica, es realitza aquesta
acció*/
if((paquet_artipprog_p->command & 0x88)==0x88)
{
reinici_valors();
}
/*En cas d'haver rebut una comanda a realitzar, s'envia un missatge de tipus
ArtIpProgReply de resposta*/
if((paquet_artipprog_p->command & 0x82)==0x82 || (paquet_artipprog_p->command &
0x84)==0x84 || (paquet_artipprog_p->command & 0x84)==0x88)
{
omplir_ipprogreply();
memcpy(buffer_udp, &paquet_ipprogreply,34);
inici_paquet_udp(ip_remota, port_artnet);
for(int i=0;i<34;i++)
{
escriure_udp(buffer_udp[i]);
}
terminar_paquet_udp();
}
}
}

/*Es llegeix l'estat del socket TCP i en funció d'aquest es procedeix*/
estat_tcp=W5100.readSnSR(1);
/*Si el socket està tancat, s'obre*/
switch(estat_tcp) {
case SnSR::CLOSED:
iniciar_tcp();
break;
/*Si s'ha establert una comunicació, es comprova si hi ha missatges al buffer de
recepció per llegir*/
case SnSR::ESTABLISHED:
mida_paquet_tcp=W5100.getRXReceivedSize(1);
if (mida_paquet_tcp > 0) {
/*Es desa el buffer de recepció*/
rebre_tcp(buffer_tcp,mida_paquet_tcp);

/*Es comprova si al missatge hi ha un mètode de HTTP, en cas afirmatiu es pasa a
generar la plana web*/
get_present=cerca_str((char *)buffer_tcp,"GET /");
post_present=cerca_str((char *)buffer_tcp,"POST /");
if (get_present >= 0 || post_present >= 0) {
/*Si hi ha el mètode POST s'interpreten les dades que inclou i es desen a les
variables corresponents*/
if(post_present>=0)
{
index= strstr ((char *)buffer_tcp,"IP=");
sscanf
(index+3,"%u.%u.%u.%u",&ip_node_web[0],&ip_node_web[1],&ip_node_web[2],&ip_node_web
[3]);
index= strstr ((char *)buffer_tcp,"SM=");
sscanf
(index+3,"%u.%u.%u.%u",&subnet_mask_node_web[0],&subnet_mask_node_web[1],&subnet_ma
sk_node_web[2],&subnet_mask_node_web[3]);
index= strstr ((char *)buffer_tcp,"GW=");
sscanf
(index+3,"%u.%u.%u.%u",&gateway_node_web[0],&gateway_node_web[1],&gateway_node_web[
2],&gateway_node_web[3]);
index= strstr ((char *)buffer_tcp,"MAC=");
```

```
sscanf
(index+4,"%X%3A%X%3A%X%3A%X%3A%X%3A%X",&mac_node_web[0],&mac_node_web[1],&mac_
node_web[2],&mac_node_web[3],&mac_node_web[4],&mac_node_web[5]);
index= strstr ((char *)buffer_tcp,"NET=");
sscanf (index+4,"%u",&net_web);
if(net_web<0)net_web=0;
if(net_web>127)net_web=127;
index= strstr ((char *)buffer_tcp,"SNET=");
sscanf (index+5,"%u",&subnet_web);
if(subnet_web<0)subnet_web=0;
if(subnet_web>15)subnet_web=15;
index= strstr ((char *)buffer_tcp,"UNI0=");
sscanf (index+5,"%u",&univers0_web);
if(univers0_web<0)univers0_web=0;
if(univers0_web>15)univers0_web=15;
index= strstr ((char *)buffer_tcp,"UNI1=");
sscanf (index+5,"%u",&univers1_web);
if(univers1_web<0)univers1_web=0;
if(univers1_web>15)univers1_web=15;
index= strstr ((char *)buffer_tcp,"NOM=");
sscanf (index+4,"%18s",nom_web);

for(int i=0;i<18;i++)
{
if(nom_web[i]==43)
nom_web[i]=32;
}
/*Sí s'ha premut la checkbox de efectuar canvis, s'actualitza l'EEPROM*/
if(cerca_str((char *)buffer_tcp,"canvis=si")>=0)
{
actualitzar_eeprom(0);
}
}
/*Es genera la plana web i es va desant en el buffer de transmisió*/
strcpy_P((char *)buffer_tcp,PSTR("HTTP/1.0 200 OK\r\nContent-Type:
text/html\r\n\r\n<!DOCTYPE HTML>\r\n"));

strcat_P((char *)buffer_tcp,PSTR("<html><form method=""post"">\r\n <h1>Configuració
del node Artnet-DMX</h1> <p><b>Realitzar canvis?</b>"));

strcat_P((char *)buffer_tcp,PSTR("<input type=""checkbox"" name=""canvis""
value=""si""></p><hr>\r\nIP actual: \r\n"));
formatejar_addr_8(ip_node,4);
strcat_P((char *)buffer_tcp,PSTR("\r\n<br><b>IP</b> nova: <b>"));
formatejar_addr_16(ip_node_web,4);
strcat_P((char *)buffer_tcp,PSTR("</b><form method=""post"">\r\n<p>IP:<input
type=""text"" name=""IP"" maxlength=""15""></p><hr>"));

strcat_P((char *)buffer_tcp,PSTR("\r\nSubnet Mask actual: "));
formatejar_addr_8(subnet_mask_node,4);
strcat_P((char *)buffer_tcp,PSTR("<br>\r\n<b>Subnet Mask</b> nova: <b>"));
formatejar_addr_16(subnet_mask_node_web,4);
strcat_P((char *)buffer_tcp,PSTR("</b><form method=""post"">\r\n<p>Subnet
Mask:<input type=""text"" name=""SM"" maxlength=""15""></p><hr>"));

strcat_P((char *)buffer_tcp,PSTR("\r\nDefault Gateway actual: "));
formatejar_addr_8(gateway_node,4);
strcat_P((char *)buffer_tcp,PSTR("<br>\r\n<b>Default Gateway</b> nova: <b>"));
formatejar_addr_16(gateway_node_web,4);
strcat_P((char *)buffer_tcp,PSTR("</b><form method=""post"">\r\n<p>Default
Gateway:<input type=""text"" name=""GW"" maxlength=""15""></p><hr>"));

strcat_P((char *)buffer_tcp,PSTR("MAC actual: "));
formatejar_addr_8(mac_node,6);
strcat_P((char *)buffer_tcp,PSTR("<br>\r\n<b>MAC</b> nova: <b>"));
formatejar_addr_16(mac_node_web,6);
strcat_P((char *)buffer_tcp,PSTR("</b><form method=""post"">\r\n<p>MAC:<input
type=""text"" name=""MAC"" maxlength=""17""></p><hr>"));
```

```
strcat_P((char *)buffer_tcp,PSTR("\r\nNet actual: "));
formatejar_addr_8(&net,1);
strcat_P((char *)buffer_tcp,PSTR("<br>\r\n<b>Net</b> nova: <b>"));
formatejar_addr_16(&net_web,1);
strcat_P((char *)buffer_tcp,PSTR("</b><form method=""post"">\r\n<p>Net (0-
127):<input type=""text"" name=""NET"" maxlength=""2""></p><hr>"));

strcat_P((char *)buffer_tcp,PSTR("\r\nSubNet actual: "));
formatejar_addr_8(&subnet,1);
strcat_P((char *)buffer_tcp,PSTR("<br>\r\n<b>SubNet</b> nova: <b>"));
formatejar_addr_16(&subnet_web,1);
strcat_P((char *)buffer_tcp,PSTR("</b><form method=""post"">\r\n<p>Subnet (0-
15):<input type=""text"" name=""SNET"" maxlength=""2""></p><hr>"));

strcat_P((char *)buffer_tcp,PSTR("\r\nUnivers port 0 actual: \r\n"));
formatejar_addr_8(&univers0,1);
strcat_P((char *)buffer_tcp,PSTR("<br>\r\n<b>Univers port 0</b> nou: <b>"));
formatejar_addr_16(&univers0_web,1);
strcat_P((char *)buffer_tcp,PSTR("</b><form method=""post"">\r\n<p>Univers p0 (0-
15):<input type=""text"" name=""UNI0"" maxlength=""2""></p><hr>"));

strcat_P((char *)buffer_tcp,PSTR("\r\nUnivers port 1 actual: \r\n"));
formatejar_addr_8(&univers1,1);
strcat_P((char *)buffer_tcp,PSTR("<br>\r\n<b>Univers port 1</b> nou: <b>"));
formatejar_addr_16(&univers1_web,1);
strcat_P((char *)buffer_tcp,PSTR("</b><form method=""post"">\r\n<p>Univers p1 (0-
15):<input type=""text"" name=""UNI1"" maxlength=""2""></p><hr>"));

strcat_P((char *)buffer_tcp,PSTR("\r\nNom actual: "));
strcat((char *)buffer_tcp,nom);
strcat_P((char *)buffer_tcp,PSTR("<br>\r\n<b>Nom</b> nou: <b>"));
strcat((char *)buffer_tcp,nom_web);
strcat_P((char *)buffer_tcp,PSTR("</b><form method=""post"">\r\n<p>Nom:<input
type=""text"" name=""NOM"" maxlength=""18""></p><hr>"));

strcat_P((char *)buffer_tcp,PSTR("<input type=""submit"" value=""Enviar""></br>"));
strcat_P((char *)buffer_tcp,PSTR("\r\n</html>\r\n"));

/*Un cop omplert el buffer amb la plana web, s'envia*/
enviar_tcp(buffer_tcp,strlen((char *)buffer_tcp));

}

/*Es finalitza la connexió TCP*/
disconnect(1);
}
break;

/*si es troba en qualsevol altre estat, es tanca el socket*/
case SnSR::FIN_WAIT:
case SnSR::CLOSING:
case SnSR::TIME_WAIT:
case SnSR::CLOSE_WAIT:
case SnSR::LAST_ACK:
close(1);
break;
}
}
}


/*Funció que comprova la validesa dels missatges Art-Net i retorna el tipus del
missatge*/
int comprovar_tipus()
{
/*Es desa el contingut del buffer de recepció*/
llegir_udp(buffer_udp,530);
```

```
/*Comprova si el missatge comença amb el missatge Art-Net/0*/
if (!memcmp(buffer_udp, artnet_inici, 8))
{
/*Si ho fa, comprova que la versió del protocol es l'adequada (0x000E)*/
if ((((buffer_udp[10] << 8) & 0xff00) | (buffer_udp[11] & 0x00FF))==0x000E)
{
/*retorna el tipus del missatge en cas afirmatiu*/
return (((buffer_udp[9] << 8) & 0xff00) | (buffer_udp[8] & 0x00FF));
}
}
return 0;
}


/*Funció que genera un missatge de tipus ArtPollReply fent servir les variables de
configuració del node i l'estructura del missatge*/
void omplir_pollreply ()
{
paquet_pollreply_p=&paquet_pollreply;
memcpy (paquet_pollreply_p->id, artnet_inici, sizeof(paquet_pollreply_p->id));
paquet_pollreply_p->opcode=codi_pollreply;
memcpy (paquet_pollreply_p->ip, ip_node, sizeof(paquet_pollreply_p->ip));
paquet_pollreply_p->port=port_artnet;
paquet_pollreply_p->verh=0;
paquet_pollreply_p->ver=0;
paquet_pollreply_p->subh=net;
paquet_pollreply_p->sub=subnet;
paquet_pollreply_p->oemh=0x00;
paquet_pollreply_p->oem=0xFF;
paquet_pollreply_p->ubea=0;
paquet_pollreply_p->status1=0xE0;
memset (paquet_pollreply_p->etsaman, 0xFF, sizeof(paquet_pollreply_p->etsaman));
memcpy (paquet_pollreply_p->shortname, nom, sizeof(paquet_pollreply_p->shortname));
memset (paquet_pollreply_p->longname, 0, sizeof(paquet_pollreply_p->longname));
memset (paquet_pollreply_p->nodereport, 0, sizeof(paquet_pollreply_p->nodereport));
paquet_pollreply_p->numbportsh=0;
paquet_pollreply_p->numbports=2;
memset (paquet_pollreply_p->porttypes, 0x80, 2);
memset (paquet_pollreply_p->porttypes+2, 0x00, 2);
memset (paquet_pollreply_p->goodinput, 0x08, sizeof(paquet_pollreply_p-
>goodinput));
paquet_pollreply_p->goodoutput[0]=0x80+estat_merge_0;
paquet_pollreply_p->goodoutput[1]=0x80+estat_merge_1;
memset (paquet_pollreply_p->goodoutput+2, 0x00, 2);
paquet_pollreply_p->swout[0]=univers0;
paquet_pollreply_p->swout[1]=univers1;
paquet_pollreply_p->swvideo=0;
paquet_pollreply_p->swmacro=0;
paquet_pollreply_p->swremote=0;
paquet_pollreply_p->sp1=0;
paquet_pollreply_p->sp2=0;
paquet_pollreply_p->sp3=0;
paquet_pollreply_p->style=0;
memcpy (paquet_pollreply_p->mac, mac_node, sizeof(paquet_pollreply_p->mac));
memset (paquet_pollreply_p->bindip, 0x00, sizeof(paquet_pollreply_p->bindip));
paquet_pollreply_p->bindindex=0;
paquet_pollreply_p->status2=0x09;
memset (paquet_pollreply_p->filler, 0x00, sizeof(paquet_pollreply_p->filler));
}

/*Funció que genera un missatge de tipus ArtIpProgReply fent servir les variables
de configuració del node i l'estructura del missatge*/
void omplir_ipprogreply()
{
paquet_ipprogreply_p=&paquet_ipprogreply;
memcpy (paquet_ipprogreply_p->id, artnet_inici, sizeof(paquet_ipprogreply_p->id));
paquet_ipprogreply_p->opcode=codi_ipprogreply;
paquet_ipprogreply_p->protverhi=0;
```

```
paquet_ipprogreply_p->protverlo=0x0E;
paquet_ipprogreply_p->progip3=ip_node[0];
paquet_ipprogreply_p->progip2=ip_node[1];
paquet_ipprogreply_p->progip1=ip_node[2];
paquet_ipprogreply_p->progip0=ip_node[3];
paquet_ipprogreply_p->progsm3=subnet_mask_node[0];
paquet_ipprogreply_p->progsm2=subnet_mask_node[1];
paquet_ipprogreply_p->progsm1=subnet_mask_node[2];
paquet_ipprogreply_p->progsm0=subnet_mask_node[3];
paquet_ipprogreply_p->progporthi=net;
paquet_ipprogreply_p->progportlo=(subnet<<4)+univers0;
}



/*Funció que inicia la transmisió d'una trama DMX al port USART0*/
void envia_dmx_0() {
estat_dmx_0 = 1;
/*La transmisió s'inicia a una freqüència més baixa (80.000kHz) per generar el
senyal de break*/
UBRR0H = ((F_CPU/80000/16) - 1) >> 8;
UBRR0L = ((F_CPU/80000/16) - 1);
/*S'habilita la interrupció de transmisió complerta*/
UCSR0B |= 1<<TXCIE0;
/*S'envia un byte buit per genera el senyal de break*/
UDR0 = 0;
}

/*Funció que inicia la transmisió d'una trama DMX al port USART1*/
void envia_dmx_1() {
estat_dmx_1 = 1;
UBRR1H = ((F_CPU/80000/16) - 1) >> 8;
UBRR1L = ((F_CPU/80000/16) - 1);
UCSR1B |= 1<<TXCIE1;
UDR1 = 0;
}

/*Interrupció que es genera cada cop que la transmisió d'un byte es compleix*/
ISR(USART0_TX_vect) {
/*Es comprova si està en estat d'enviar totes les dades DMX primer ja que es
l'estat més freqüent*/
if (estat_dmx_0 == 3) {
/*S'envia la resta de dades de la trama DMX*/
UDR0 = buffer_dmx0[ptr_dmx_0];
ptr_dmx_0++;
if (ptr_dmx_0 >= 512) {
estat_dmx_0 = 4;
}
}
else if (estat_dmx_0 == 1) {
/*Es configura el baudrate a 250.000 kHz*/
UBRR0H = ((F_CPU/250000/16) - 1) >> 8;
UBRR0L = ((F_CPU/250000/16) - 1);
/*S'envia el byte d'start*/
UDR0 = 0;
estat_dmx_0 = 2;
}
else if (estat_dmx_0 == 2) {
/*S'envia el primer byte de dades DMX i s'inicialitza el punter*/
UDR0 = buffer_dmx0[0];
ptr_dmx_0 = 1;
estat_dmx_0 = 3;
}
else {
/*Un cop enviats tots el bytes, es desabilita l'interrupció, es posa a 0 el flag
d'aquesta i la variable d'estat de transmisió de DMX*/
UCSR0B &= ~(1<<TXCIE0);
UCSR0A |= 1<<TXC0;
```

```
estat_dmx_0 = 0;
}
}

ISR(USART1_TX_vect) {
if (estat_dmx_1 == 3) {
UDR1 = buffer_dmx1[ptr_dmx_1];
ptr_dmx_1++;
if (ptr_dmx_1 >= 512) {
estat_dmx_1 = 4;
}
}
else if (estat_dmx_1 == 1) {
UBRR1H = ((F_CPU/250000/16) - 1) >> 8;
UBRR1L = ((F_CPU/250000/16) - 1);
UDR1 = 0;
estat_dmx_1 = 2;
}
else if (estat_dmx_1 == 2) {
UDR1 = buffer_dmx1[0];
ptr_dmx_1 = 1;
estat_dmx_1 = 3;
}
else {
UCSR1B &= ~(1<<TXCIE1);
UCSR1A |= 1<<TXC1;
estat_dmx_1 = 0;
}
}

/*Funció d'inicialització dels registres dels ports USART. Es configuren amb 8 bits
de dades i 2 bits de stop*/
void iniciar_usart()
{
UBRR0H = ((F_CPU/250000/16) - 1) >> 8;
UBRR0L = ((F_CPU/250000/16) - 1);
UCSR0A = 1<<UDRE0;
UCSR0C = 1<<USBS0 | 1<<UCSZ01 | 1<<UCSZ00; // 2 stop bits,8 data bitss
UCSR0B = 1<<TXEN0; // turn it on

UBRR1H = ((F_CPU/250000/16) - 1) >> 8;
UBRR1L = ((F_CPU/250000/16) - 1);
UCSR1A = 1<<UDRE1;
UCSR1C = 1<<USBS1 | 1<<UCSZ11 | 1<<UCSZ10; // 2 stop bits,8 data bitss
UCSR1B = 1<<TXEN1; // turn it on
}

/*Inicialització del xip w5100*/
void iniciar_w5100(uint8_t *mac, uint8_t *ip_address, uint8_t *gateway, uint8_t
*mask)
{
W5100.init();
W5100.setMACAddress(mac);
W5100.setIPAddress(ip_address);
W5100.setGatewayIp(gateway);
W5100.setSubnetMask(mask);
}

/*Inicialització del socket UDP*/
void iniciar_udp(uint16_t port)
{
socket(0, SnMR::UDP, port, 0);
}

/*Funció que comprova la recepció de paquets UDP i en cas afirmatiu desa la IP font
i retorna la mida del paquet*/
int extreure_udp()
{
```

```
/*Es descarten bytes pertanyents al paquet anteriorment rebut*/
flush_udp();

if (W5100.getRXReceivedSize(0) > 0)
{
uint8_t tmpBuf[8];
int ret =0;
/*Llegeix els 8 primer bytes del paquet UDP, capçalera que posa el xip amb la
direcció IP font, el port destí i la mida del paquet*/
ret = recv(0,tmpBuf,8);
if (ret > 0)
{
memcpy(ip_remota,tmpBuf,4);
/*Desa la mida del paquet i la retorna*/
restant = tmpBuf[6];
restant = (restant << 8) + tmpBuf[7];
ret = restant;
/*la variable d'identificació serà 0 al iniciar el programa i quan passin 10 segons
al mode merge
desde el moment en que es va rebre l'últim paquet d'una de les dues IP*/
}
return ret;
}
return 0;
}

/*En cas de que quedin bytes per llegir d'un paquet anterior, aquesta funció els
llegeix per tal d'actualitzar el punter de recepció del xip i fer
que apunti a la direcció del nou paquet rebut*/
void flush_udp()
{
while (restant)
{
llegir_udp();
}
}

/*Funció que llegeix el buffer de recepció UDP byte per byte*/
int llegir_udp()
{
uint8_t byte;

if ((restant > 0) && (recv(0, &byte, 1) > 0))
{
// We read things without any problems
restant--;
return byte;
}

// If we get here, there's no data available
return -1;
}

/*Funció que llegeix un número de bytes especificat del buffer de recepció UDP i
els desa en el array especificat*/
int llegir_udp(unsigned char* buffer, size_t len)
{

if (restant > 0)
{
int got;
if (restant <= len)
{
got = recv(0, buffer, restant);
}
else
{
got = recv(0, buffer, len);
```

```
}

if (got > 0)
{
restant -= got;
return got;
}
}
return -1;
}

/*Funció per preparar la transmisió d'un paquet UDP. Escriu en els registres
indicats la IP i port destí*/
int inici_paquet_udp(uint8_t *ip, uint16_t port)
{
offset = 0;
return startUDP(0, ip, port);
}

/*Funció que escriu la comanda d'enviar el paquet UDP al xip d'ethernet*/
int terminar_paquet_udp()
{
return sendUDP(0);
}

/*Funció que escriu un byte al buffer de transmisió UDP*/
size_t escriure_udp(uint8_t byte)
{
return escriure_udp(&byte, 1);
}

/*Funció que passa els continguts d'un array al buffer de transmisió UDP*/
size_t escriure_udp(const uint8_t *buffer, size_t size)
{
uint16_t bytes_written = bufferData(0, offset, buffer, size);
offset += bytes_written;
return bytes_written;
}

/*Funció que obre el socket TCP*/
void iniciar_tcp()
{
socket(1, SnMR::TCP, 80, 0);
listen(1);
}

/*Funció per transmetre bytes via el socket TCP*/
void enviar_tcp(const uint8_t *buffer_tcp,uint16_t buflen)
{
uint16_t ptr,realaddr,txsize,timeout;

txsize= W5100.readSnTX_FSR(1);
timeout=0;
while (txsize < buflen) {
_delay_ms(1);

txsize= W5100.readSnTX_FSR(1);

/*si al cap d'un segon el buffer de transmisió no te prou espai per el paquet, es
tanca la connexió*/
if (timeout++ > 1000) {
disconnect(1);
}
}
/*Es calcula la posició del punter del buffer de transmisió*/
ptr = W5100.readSnTX_WR(1);
while(buflen) {
buflen--;
```

```
realaddr = 0x4800 + (ptr & 0x07FF);
/*S'escriu byte a byte el contingut del paquet al buffer de transmisió*/
W5100.write(realaddr,*buffer_tcp);
ptr++;
buffer_tcp++;
}
/*Es dóna la comanda de transmisió al xip d'ethernet*/
W5100.writeSnTX_WR(1, ptr);
W5100.execCmdSn(1, Sock_SEND);

}

/*Funció per rebre bytes del socket TCP*/
void rebre_tcp(uint8_t *buffer_tcp,uint16_t buflen)
{
uint16_t ptr,realaddr;
/*Si el paquet té una mida superior a 1000 bytes, es trunca*/
if (buflen > 1000)
buflen=1000 - 2;
ptr = W5100.readSnRX_RD(1);
/*Es calcula la posició del punter del buffer de recepció*/
while(buflen) {
buflen--;
realaddr= 0x6800 + (ptr & 0x07FF);
*buffer_tcp = W5100.read(realaddr);
ptr++;
buffer_tcp++;
}
/*Es finalitza el paquet rebut amb un caràcter nul*/
*buffer_tcp='\0';
W5100.writeSnRX_RD(1, ptr);
/*Es dóna la comanda de recepció complerta al xip d'ethernet*/
W5100.execCmdSn(1, Sock_RECV);
}


/*Funció que cerca un string en un altre. Si el troba retorna 0 o un número
positiu*/
int cerca_str(char *s,char *t)
{
uint16_t i,n;

n=strlen(t);
for(i=0;*(s+i); i++) {
if (strncmp(s+i,t,n) == 0)
return i;
}
return -1;
}

/*Funcio que actualitza les dades de les variables de la web*/
void iniciar_dades_web()
{
eeprom_read_block((void*) ip_node, ip_node_eeprom, 4);
eeprom_read_block((void*) subnet_mask_node, subnet_mask_eeprom, 4);
eeprom_read_block((void*) gateway_node, gateway_node_eeprom, 4);
eeprom_read_block((void*) mac_node, mac_node_eeprom, 6);
net=eeprom_read_byte (&net_eeprom);
subnet=eeprom_read_byte (&subnet_eeprom);
univers0=eeprom_read_byte (&univers0_eeprom);
univers1=eeprom_read_byte (&univers1_eeprom);
eeprom_read_block((void*) nom, nom_eeprom, 18);

for(int i=0;i<6;i++)
{
if(i<4)
{
ip_node_web[i]=ip_node[i];
```

```
subnet_mask_node_web[i]=subnet_mask_node[i];
gateway_node_web[i]=gateway_node[i];
}
mac_node_web[i]=mac_node[i];
}
net_web=net;
subnet_web=subnet;
univers0_web=univers0;
univers1_web=univers1;
memcpy(nom_web,nom,18);
}

/*Funció que desa els valors de fàbrica a les variables del nodes i després
actualitza la EEPROM amb aquests, provocant un reinici del xip ethernet*/
void reinici_valors()
{
memcpy(ip_node,ip_node_fab,4);
memcpy(subnet_mask_node,subnet_mask_node_fab,4);
memcpy(gateway_node,gateway_node_fab,4);
memcpy(mac_node,mac_node_fab,6);
net_web=net_fab;
subnet_web=subnet_fab;
univers0_web=univers0_fab;
univers1_web=univers1_fab;
memcpy(nom_web,nom_fab,18);
actualitzar_eeprom(1);
}



/*S'actualitzen les dades de configuració del node a la EEPROM i es reseteja el xip
Ethernet per tal d'aplicar els canvis*/
/*En funció del mode, les dades a desar provenen de la variable de la web o no*/
void actualitzar_eeprom(uint8_t mode)
{
if(mode==0)
{
for(int i=0;i<6;i++)
{
if(i<4)
{
ip_node[i]=ip_node_web[i];
subnet_mask_node[i]=subnet_mask_node_web[i];
gateway_node[i]=gateway_node_web[i];
}
mac_node[i]=mac_node_web[i];
}
}
eeprom_update_block((const void*)ip_node,(void*)ip_node_eeprom,4);
eeprom_update_block((const void*)subnet_mask_node,(void*)subnet_mask_eeprom,4);
eeprom_update_block((const void*)gateway_node,(void*)gateway_node_eeprom,4);
eeprom_update_block((const void*)mac_node,(void*)mac_node_eeprom,6);
eeprom_update_byte(&net_eeprom,(uint8_t)net_web);
eeprom_update_byte(&subnet_eeprom,(uint8_t)subnet_web);
eeprom_update_byte(&univers0_eeprom,(uint8_t)univers0_web);
eeprom_update_byte(&univers1_eeprom,(uint8_t)univers1_web);
eeprom_update_block((const void*)nom_web,(void*)nom_eeprom,18);
iniciar_dades_web();
iniciar_sockets();
if(talktome==2)
{
enviar_pollreply();
}

}

/*Funció per a convertir les addreces de 8 bits a text*/
void formatejar_addr_8(unsigned char* addr, uint8_t num)
```

```
{
for(int i=0;i<num;i++)
{
if(num!=6)
{
sprintf(conversio_str,"%d",addr[i]);
}else
{
sprintf(conversio_str,"%X",addr[i]);
}
strcat((char *)buffer_tcp,conversio_str);
if(num==4)
{
if(i<3) strcat_P((char *)buffer_tcp,PSTR("."));
}
if(num==6)
{
if(i<5) strcat_P((char *)buffer_tcp,PSTR(":"));
}
}

}

/*Funció per a convertir les addreces de 16 bits a text*/
void formatejar_addr_16(uint16_t* addr, uint8_t num)
{
for(int i=0;i<num;i++)
{
if(num!=6)
{
sprintf(conversio_str,"%d",addr[i]);
}else
{
sprintf(conversio_str,"%X",addr[i]);
}
strcat((char *)buffer_tcp,conversio_str);
if(num==4)
{
if(i<3) strcat_P((char *)buffer_tcp,PSTR("."));
}
if(num==6)
{
if(i<5) strcat_P((char *)buffer_tcp,PSTR(":"));
}

}

}
/*Inicialització de tots dos sockets a utilitzar del xip ethernet*/
void iniciar_sockets()
{
iniciar_w5100(mac_node,ip_node,gateway_node,subnet_mask_node);
iniciar_udp(port_artnet);
iniciar_tcp();
}


/*Funció que envia un missatge de tipus ArtPollReply via UDP*/
void enviar_pollreply()
{
omplir_pollreply();
memcpy(buffer_udp, &paquet_pollreply,239);
for (int i=0;i<4;i++)
{
ip_remota_broad[i]=(ip_remota[i]&subnet_mask_node[i])+(~subnet_mask_node[i]);
}
inici_paquet_udp(ip_remota_broad, port_artnet);
for(int i=0;i<240;i++)
```

```
{
escriure_udp(buffer_udp[i]);
}
terminar_paquet_udp();
}

/*Rutines d'interrupció de comparació. Cada segon incrementen una variable per
determinar
si cal desactivar el mode merge*/

ISR(TIMER1_COMPA_vect)
{
segons_inactivitat_0++;
}
ISR(TIMER3_COMPA_vect)
{
segons_inactivitat_1++;
}
```

B. PAQUETS DEL PROTOCOL ART-NET

El protocol Art-Net basa la seva comunicació en diferents tipus de paquets que contenen les comandes i dades pertanyents. El llistat de paquets descrits a continuació no conformen la totalitat dels inclosos en el protocol Art-Net, sino que inclou aquells que el node es capaç d'identificar i enviar. S'ha considerat que la selecció efectuada permet al node operar de forma eficient en una xarxa Art-Net després d'avaluar les opcions que acostumen a incorporar els programes dedicats al protocol.

ArtPoll				
Camp	**Nom**	**Tamany**	**Bit**	**Descripció**
1	ID[8]	Int8	-	Array de 8 caràcters, el caràcter final es nul. Valor = 'A' 'r' 't' '-' 'N' 'e' 't' 0x00
2	OpCode	Int16	-	Codi que defineix el tipus del missatge. En aquest cas el codi es 0x2000.
3	ProtVerHi	Int8	-	Byte superior del número de revisió del protocol.
4	ProtVerLo	Int8	-	Byte inferior del número de revisió del protocol. Valor actual = 0x0014.
5	TalkToMe	Int8	-	Defineix comportament del node.
			7-4	No fet servir, transmetre com a 0.
			3	0 = Els missatges de diagnostic han de ser broadcast.
				1 = Els missatges de diagnostic han de ser unicast.
			2	0 = No enviar missatges de diagnòstic.
				1 = Enviar missatges de diagnòstic.
			1	0 = Només enviar missatges ArtPollReply en resposta a ArtPoll o ArtAddress
				1 = Enviar missatges ArtPollReply sempre que la configuració del node canvii.
			0	No fet servir, transmetre com a 0.
6	Prioritat	Int8	-	Prioritat del missatge.

Taula 6. Camps del paquet ArtPoll

ArtPollReply				
Camp	**Nom**	**Tamany**	**Bit**	**Descripció**
1	ID[8]	Int8	-	Array de 8 caràcters, el caràcter final es nul. Valor = 'A' 'r' 't' '-' 'N' 'e' 't' 0x00
2	OpCode	Int16	-	Codi que defineix el tipus del missatge. En aquest cas el codi es 0x2100.
3	IP Address[4]	Int8	-	Array de 4 caràcters que conté la IP del node.
4	Port	Int16	-	Port del node. Sempre és 0x1936.
5	VersInfoH	Int8		Bit superior de la versió del firmware.
6	VersInfo	Int8	-	Bit inferior de la versió del firmware.
7	NetSwitch	Int8	-	Bits 14-8 de l'adreça de 15 bits dels ports del node.
8	SubSwitch	Int8	-	Bits 7-4 de l'adreça de 15 bits dels ports del node.
9	OemHi	Int8	-	Bit superior del valor OEM.
10	Oem	Int8	-	Bit inferior del valor OEM. Aquest valor es utilitzat per descriure el venedor del equipament.
11	Ubea Version	Int8	-	Versió del firmware del UBEA (User Bios Extension Area).
12	Status1	Int8	-	Registre de l'estat general.
			7-6	Indicador d'estat
				00 Estat desconegut.
				01 Estat de localització.
				10 Estat mutejat.
				11 Estat normal.
			5-4	Autorització de programació dels ports.
				00 Autorització no especificada.
				01 Programació mitjançant panells.
				10 Progamació efectuada per la xarxa.
				11 Camp no utilitzat.
			3	No implementat.
			2	0 = Boot del firmware normal (flash).
				1 = Boot desde ROM.
			1	0 = Node no compatible amb RDM
				1 = Node compatible amb RDM
			0	0 = Ubea no present o corrupte.
				1 = Ubea present.

Taula 7. Camps del 1 al 12 del paquet ArtPollReply

ArtPollReply				
Camp	**Nom**	**Tamany**	**Bit**	**Descripció**
13	EstaManLo	Int8	-	Byte inferior del codi ESTA del fabricant.
14	EstaManHi	Int8	-	Byte superior del codi ESTA del fabricant.
15	ShortName[18]	Int8	-	Nom curt del node.
16	LongName[64]	Int8	-	Nom llarg del node.
17	NodeReport[64]	Int8	-	Informe de l'estat de funcionament del node.
18	NumPortsHi	Int8	-	Byte superior del número de ports del node.
19	NumPortsLo	Int8	-	Byte inferior del número de ports del node.
20	PortTypes[4]	Int8	-	Array que defineix el mode d'operació i el protocol de cada port del node.
			7	A 1 si el canal pot transmetre dades procedents de la xarxa Art-Net.
			6	A 1 si el canal pot transmetre senyals a la xarxa Art-Net.
			5-0	00000 = DMX512 00001 = MIDI 00010 = Avab 00011 = Colortran CMX 00100 = ADB 62.5 00101 = Art-Net
21	GoodInput[4]	Int8	-	Array que defineix l'estat dels ports d'entrada del node.
			7	A 1 – Dades rebudes.
			6	A 1 – El canal inclou paquets de prova DMX512.
			5	A 1 – El canal inclou paquets d'informació del sistema.
			4	A 1 – El canal inclou paquets de text DMX.
			3	A 1 – Entrada deshabilitada.
			2	A 1 – Errors de recepció detectats.
			1-0	No fets servir.
22	GoodOutput[4]	Int8	-	Array que defineix l'estat dels ports de sortida del node.
			7	A 1 – S'estan transmetent dades.
			6	A 1 – El canal inclou dades DMX512
			5	A 1 – El canal inclou paquets d'informació del sistema.
			4	A 1 – El canal inclou paquets de text DMX.

Taula 8. Camps del 13 al 22 del paquet ArtPollReply

ArtPollReply				
Camp	**Nom**	**Tamany**	**Bit**	**Descripció**
22	GoodOutput[4]	Int8	3	A 1 – La sortida està unint (merge) dades
			2	A 1 – Output short detectat al iniciar.
			1	A 1 – Mode d'unió (merge) es LTP.
			0	No fet servir
23	SwIn[4]	Int8	-	Bits 3-0 de l'adreça de 15 bits dels ports d'entrada del node.
24	SwOut[4]	Int8	-	Bits 3-0 de l'adreça de 15 bits dels ports de sortida del node.
25	SwVideo	Int8	-	A 00 si la sortida de video mostra dades locals i a 01 si les dades són d'Ethernet.
26	SwMacro	Int8	-	Si el node suporta claus macro, el byte representa el valors dels disparadors.
			7	A 1 – Macro 8 activa.
			6	A 1 – Macro 7 activa.
			5	A 1 – Macro 6 activa.
			4	A 1 – Macro 5 activa.
			3	A 1 – Macro 4 activa.
			2	A 1 – Macro 3 activa.
			1	A 1 – Macro 2 activa.
			0	A 1 – Macro 1 activa.
27	SwRemote	Int8	-	Si el node suporta entrades d'activació remota, el byte representa el valors dels disparadors.
			7	A 1 – Remote 8 activa.
			6	A 1 – Remote 7 activa.
			5	A 1 – Remote 6 activa.
			4	A 1 – Remote 5 activa.
			3	A 1 – Remote 4 activa.
			2	A 1 – Remote 3 activa.
			1	A 1 – Remote 2 activa.
			0	A 1 – Remote 1 activa.
28	Spare	Int8	-	No fet servir.
29	Spare	Int8	-	No fet servir.
30	Spare	Int8	-	No fet servir.
31	Style	Int8	-	Codi d'estil, defineix el tipus d'equipament al que pertany el node.

Taula 9. Camps del 22 al 31 del paquet ArtPollReply

ArtPollReply				
Camp	**Nom**	**Tamany**	**Bit**	**Descripció**
32	MAC Hi	Int8	-	Byte superior de l'adreça MAC.
33	MAC	Int8	-	Adreça MAC.
34	MAC	Int8	-	Adreça MAC.
35	MAC	Int8	-	Adreça MAC.
36	MAC	Int8	-	Adreça MAC.
37	MAC Lo	Int8	-	Byte inferior de l'adreça MAC.
38	BindIp[4]	Int8	-	Si el dispositiu forma part d'un producte modul·lar, aquesta es l'IP del dispositiu principal.
39	BindIndex	Int8	-	A 0 en cas de no haver vinclament, en qualsevol altre cas representa el número de dispositius vinclats.
40	Status2	Int8	0	A 1 – El dispositiu permet configurar-se via plana web.
			1	0 = IP del node configurada manualment.
				1 = IP obtinguda amb servidor DHCP.
			2	0 = Dispositiu no compatible amb DHCP.
				1 = Dispositiu compatible amb DHCP.
			3	0 = El node suporta adreces de port de 8 bits (Art-Net II)
				1 = El node suporta adreces de port de 15 bits (Art-Net 3)
41	Filler	26 x Int8		No utilitzats, transmetre com a 0.

Taula 10. Camps del 32 al 41 del paquet ArtPollReply

ArtIpProg				
Camp	**Nom**	**Tamany**	**Bit**	**Descripció**
1	ID[8]	Int8	-	Array de 8 caràcters, el caràcter final es nul. Valor = 'A' 'r' 't' '-' 'N' 'e' 't' 0x00
2	OpCode	Int16	-	Codi que defineix el tipus del missatge. En aquest cas el codi es 0xF800.
3	ProtVerHi	Int8	-	Byte superior del número de revisió del protocol.
4	ProtVerLo	Int8	-	Byte inferior del número de revisió del protocol. Valor actual = 0x0014.

Taula 11. Camps del 1 al 4 del paquet ArtIpProg

ArtIpProg				
Camp	**Nom**	**Tamany**	**Bit**	**Descripció**
5	Filler1	Int8	-	No utilitzat, transmetre com a 0.
6	Filler2	Int8	-	No utilitzat, transmetre com a 0.
7	Command	Int8	-	Defineix el tipus d'acció del paquet.
			7	A 1 – Habilita la programació del node.
			6	A 1 – Habilita l'us de DHCP.
			5-4	No fets servir
			3	A 1 – Retorna paràmetres als valors inicials.
			2	A 1 – Programa l'adreça IP
			1	A 1 – Programa la màscara de subxarxa.
			0	A 1 –Programa el port UDP.
8	Filler4	Int8	-	No utilitzat, transmetre com a 0.
9	ProgIpHi	Int8	-	Byte superior de l'IP a programar
10	ProgIp2	Int8	-	Adreça IP a programar.
11	ProgIp1	Int8	-	Adreça IP a programar.
12	ProgIpLo	Int8	-	Byte inferior de l'IP a programar.
13	ProgSmHi	Int8	-	Byte superior de la màscara de subxarxa a programar.
14	ProgSm2	Int8	-	Màscara de subxarxa a programar.
15	ProgSm1	Int8	-	Màscara de subxarxa a programar.
16	ProgSmLo	Int8	-	Byte inferior de la màscara de subxarxa a programar.
17	ProgPort Hi	Int8	-	Byte superior del port a programar.
18	ProgPort Lo	Int8	-	Byte inferior del port a programar.
16-26	Spare 1-8	Int8 I	-	No utilitzats, transmetre com a 0.

Taula 12. Camps del 5 al 26 del paquet ArtIpProg

ArtIpProgReply				
Camp	**Nom**	**Tamany**	**Bit**	**Descripció**
1	ID[8]	Int8	-	Array de 8 caràcters, el caràcter final es nul. Valor = 'A' 'r' 't' '-' 'N' 'e' 't' 0x00
2	OpCode	Int16	-	Codi que defineix el tipus del missatge. En aquest cas el codi es 0xF900.
3	ProtVerHi	Int8	-	Byte superior del número de revisió del protocol.
4	ProtVerLo	Int8	-	Byte inferior del número de revisió del protocol. Valor actual = 0x0014.

Taula 13. Camps del 1 al 4 del paquet ArtIpProgReply

ArtIpProgReply				
Camp	**Nom**	**Tamany**	**Bit**	**Descripció**
5	Filler1	Int8	-	No utilitzat, transmetre com a 0.
6	Filler2	Int8	-	No utilitzat, transmetre com a 0.
7	Filler3	Int8	-	No utilitzat, transmetre com a 0.
8	Filler4	Int8	-	No utilitzat, transmetre com a 0.
9	ProgIpHi	Int8	-	Byte superior de l'IP a programar
10	ProgIp2	Int8	-	Adreça IP programada.
11	ProgIp1	Int8	-	Adreça IP programada.
12	ProgIpLo	Int8	-	Byte inferior de l'IP programada.
13	ProgSmHi	Int8	-	Byte superior de la màscara de subxarxa programada.
14	ProgSm2	Int8	-	Màscara de subxarxa programada.
15	ProgSm1	Int8	-	Màscara de subxarxa programada.
16	ProgSmLo	Int8	-	Byte inferior de la màscara de subxarxa programada.
17	ProgPort Hi	Int8	-	Byte superior del port programat.
18	ProgPort Lo	Int8	-	Byte inferior del port programat.
19	Status	Int8	7	No utilitzat.
			6	DHCP habilitat.
			5-0	No utilitzat.
20-26	Spare 1-7	Int8	-	No utilitzats, transmetre com a 0.

Taula 14. Camps del 5 al 26 del paquet ArtIpProgReply

ArtAddress				
Camp	**Nom**	**Tamany**	**Bit**	**Descripció**
1	ID[8]	Int8	-	Array de 8 caràcters, el caràcter final es nul. Valor = 'A' 'r' 't' '-' 'N' 'e' 't' 0x00
2	OpCode	Int16	-	Codi que defineix el tipus del missatge. En aquest cas el codi es 0xF900.
3	ProtVerHi	Int8	-	Byte superior del número de revisió del protocol.
4	ProtVerLo	Int8	-	Byte inferior del número de revisió del protocol. Valor actual = 0x0014.
5	NetSwitch	Int8	-	Bits 14-8 de l'adreça de 15 bits dels ports del node.
6	Filler2	Int8	-	No utilitzats.

Taula 15. Camps del 1 al 6 del paquet ArtAddress

ArtAddress				
Camp	**Nom**	**Tamany**	**Bit**	**Descripció**
7	Short Name[18]	Int8	-	Nom curt del node.
8	Long Name[64]	Int8	-	Nom llarg del node.
9	SwIn[4]	Int8	-	Bits 3-0 de l'adreça de 15 bits dels ports d'entrada del node.
10	SwOut[4]	Int8	-	Bits 3-0 de l'adreça de 15 bits dels ports de sortida del node.
11	SubSwitch	Int8	-	Bits 7-4 de l'adreça de 15 bits del ports del node.
12	SwVideo	Int8	-	No utilitzat.
13	Command	Int8	-	Comandes de configuració del node

Valor	Acció
0x00	Sense acció.
0x00	Cancel·la el mode merge.
0x00	Comportament normal dels indicadors lumínics d'estat del node.
0x00	Els indicadors lumínics del node s'apaguen.
0x00	Flash ràpid dels indicadors lumínics del node.
0x00	Reseteig dels flags d'error i altres del node.
0x00	Merge del port 0 DMX en mode LTP.
0x00	Merge del port 1 DMX en mode LTP.
0x00	Merge del port 2 DMX en mode LTP.
0x00	Merge del port 3 DMX en mode LTP.
0x00	Merge del port 0 DMX en mode HTP.
0x00	Merge del port 1 DMX en mode HTP.
0x00	Merge del port 2 DMX en mode HTP.
0x00	Merge del port 3 DMX en mode HTP.
0x00	Esborra el buffer d'output DMX del port 0.
0x00	Esborra el buffer d'output DMX del port 1.
0x00	Esborra el buffer d'output DMX del port 2.
0x00	Esborra el buffer d'output DMX del port 3.

Taula 16. Camps del 7 al 13 del paquet ArtAddress

ArtDMX				
Camp	**Nom**	**Tamany**	**Bit**	**Descripció**
1	ID[8]	Int8	-	Array de 8 caràcters, el caràcter final es nul. Valor = 'A' 'r' 't' '-' 'N' 'e' 't' 0x00
2	OpCode	Int16	-	Codi que defineix el tipus del missatge. En aquest cas el codi es 0x5000.
3	ProtVerHi	Int8	-	Byte superior del número de revisió del protocol.
4	ProtVerLo	Int8	-	Byte inferior del número de revisió del protocol. Valor actual = 0x0014.
5	Sequence	Int8		Número de seqüència de la transmissió.
6	Physical	Int8	-	Port físic del que prové el paquet.
7	SubUni	Int8	-	Byte inferior de l'adreça de 15 bits del port del node.
8	Net	Int8	-	7 bits superiors de l'adreça de 15 bits del port del node.
9	LenghtHi	Int8	-	Byte superior de la longitud de la trama DMX512 continguda.
10	Lenght	Int8	-	Byte inferior de la longitud de la trama DMX512 continguda.
11	Data[Length]	Int8	-	Dades dels canals DMX512.

Taula 17. Camps del 1 al 11 del paquet ArtDMX

C. CÀLCULS

Per tal de mantenir el cost per unitat del node baix s'ha optat per fer ús de reguladors de tensió lineals. La seva baixa eficiència significa que poden arribar a dissipar una gran quantitat d'energia, essent proclius a sobreescalfar-se. Aquest fet ens obliga a comprovar si es necessària la instal·lació d'un dissipador en qualsevol dels dos reguladors, i així garantitzar que el xip operarà dins dels marges de temperatura establerts pel fabricant.

El primer pas per a realitzar aquesta comprovació es determinar quina és l'intensitat màxima que pot arriba a passar pels reguladors. Aquesta s'estima de 394mA pel regulador de 5V i de 185mA pel regulador de 3.3V.

Component	Consum màxim estimat (mA)
Atmega 1284p	50
LED x 3	39
WIZ812MJ	185
SN75176 x 2	120

Taula 18. Consum estimat dels components

Un cop feta aquesta estimació, s'ha de calcular quina potència dissiparà cada regulador. Es pretén que el node pugui ser alimentat amb tensions d'entre 7 i 12 volts, per tant els càlculs pel regulador de 5 volts es faran amb la tensió més elevada possible d'alimentació, es a dir, 12V (cas en que més energia es dissipada). En el cas del regulador de 3.3V, aquest s'alimentarà a partir de la tensió generada pel regulador de 5V, així que aquesta serà la seva tensió d'entrada. Fent servir l'eqüació 1, calculem la potència dissipada per tots dos reguladors.

$$P_D = P_I \times P_O = (V_I \times I) - (V_O \times I) \qquad \text{(eq.1)}$$

Essent:

P_D= Potència dissipada al regulador.

V_I= Voltatge d'entrada al regulador.

V_O= Voltatge de sortida del regulador.

I= Intensitat que circula pel regulador.

Trobem que la potència dissipada pel regulador de 5V es de 2.76W i la potència dissipada pel regulador de 3.3V es de 0.31W. Un cop trobada la potència cal determinar la resistència tèrmica màxima necessària per dissipar-la, mantenint la temperatura del xip als límits establerts pel fabricant. Aquest càlcul es realitzarà fent ús de l'eqüació 2.

$$\theta_{JA} = \frac{(T_J - T_A)}{P_D} \qquad \text{(eq.2)}$$

Essent:

θ_{JA}= Resistència tèrmica entre unió (interior del xip) i ambient.

T_J= Temperatura a l'interior del xip (unió).

T_A= Temperatura ambient.

P_D= Potència dissipada al regulador.

La temperatura màxima d'unió establerta a la fulla d'especificacions de tots dos xips es de 150ºC, i suposarem una temperatura ambient de 50ºC. La resistència tèrmica calculada amb aquests paràmetres es de 36.23ºC/W pel regulador de 5V i de 322.58ºC/W pel regulador de 3.3V.

La resistència tèrmica entre unió i ambient que garantitza l'encapsulat TO-220 es de 50ºC/W, més elevada que la màxima calculada pel regulador de 5V, així que per tant farà falta l'us d'un dissipador en aquest.

La resistència tèrmica entre unió i encapsulat al xip es de 3ºC/W, així que farà falta que el dissipador tingui com a màxim una resistència tèrmica (encapsulat-ambient) de 33.23ºC/W. El dissipador escollit proporciona una resistència tèrmica encapsulat-ambient de 24ºC/W, així que compleix amb els requisits tèrmics.