UNIVERSIDADE DA CORUÑA

ESCOLA UNIVERSITARIA POLITÉCNICA

Grao en Enxeñaría Electrónica Industrial e Automática

TRABALLO DE FIN DE GRAO

TFG Nº: **770G01A98**

TÍTULO: **DESEÑO DUNHA REDE SEN FÍOS DE TELEMEDIDA**

AUTOR: **VÍCTOR MANUEL ANIDOS BLANCO**

TUTOR: **JUAN MANUEL RIVAS RODRÍGUEZ**

DATA: **FEBREIRO DE 2016**

Fdo.: O AUTOR

Fdo.: O TUTOR

TÍTULO: **DESEÑO DUNHA REDE SEN FÍOS DE TELEMEDIDA**

ÍNDICE XERAL

PETICIONARIO: **ESCOLA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

DATA: FEBREIRO DE 2016

AUTOR: O ALUMNO

Fdo.: VÍCTOR MANUEL ANIDOS BLANCO

Índice de figuras

Índice de taboas

Listado de códigos de programación

TÍTULO: **DESEÑO DUNHA REDE SEN FÍOS DE TELEMEDIDA**

MEMORIA

PETICIONARIO: **ESCOLA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

DATA: FEBREIRO DE 2016

AUTOR: O ALUMNO

Fdo.: VÍCTOR MANUEL ANIDOS BLANCO

Índice do documento MEMORIA

1 Obxecto

O presente documento ten como obxecto o deseño dunha rede sen fíos para medidas a distancia, contemplando as posibles solucións que aporta o mercado e buscando a máis óptima á hora da súa realización.

Terase en conta a compatibilidade coa plataforma Arduino, que será a o soporte do deseño, e sobre a cal se baseará toda a programación.

Este traballo de fin de grao documentará tanto o software que se implementará sobre Arduino, como as posibles achegas hardware que se podan desenrolar ao longo do deseño, xa sexa por comodidade de conexión ou por necesidade imperiosa do mesmo.

Entre os temas a profundar está:

- A forma de establecer a topoloxía de rede.
- Utilización dun modo repetición para conectar os nodos máis afastados do módulo mestre.
- O seguimento da fonte de enerxía.
- A forma de enviar a información.
- A posibilidade dun modo de aforro de enerxía.
- A posibilidade de cifrar a información saínte.

Deberase realizar un dimensionado con marxes suficientes para a protección tanto dos aparellos electrónicos, como das persoas, respectando a lexislación e normas de aplicación sobre este ámbito.

Estudarase a posibilidade de incorporación de solucións existentes no mercado ao deseño, de forma que a súa compatibilidade sexa ampla e a súa utilización poda ampliarse a outros proxectos de forma rápida e funcional.

Como obxecto final estará a posible incorporación do deseño e do prototipo saído del, a unha rede de medida industrial.

2 Alcance

A finalidade última deste documento e plasmar de forma teórica e xustificada a realización e o modo de funcionamento dun prototipo real. Mediante cálculos teóricos, simulados e empíricos farase o dimensionado do modelo para buscar o funcionamento óptimo do mesmo.

A través dos planos e esquemáticos aportados, documentarse o proceso de realización do prototipo e a súa posta en funcionamento nas condicións necesarias e futuras. Nel incluirase tanto a parte do hardware (realización de placas de circuíto impreso, conexionado de placas existentes, adaptacións de módulos comerciais aos requisitos), como a parte software.

A base de todo o prototipo será a plataforma Arduino, polo que se utilizará a linguaxe de programación propia da marca. Esta plataforma é aberta e susceptible de poder ser modificada por calquera persoa interesada.

Durante o proceso de análise de solucións, e en base a novas necesidades que aparecen no desenrolo do modelo, é posible que se inclúan novas funcionalidades que non se teñen en conta nos requisitos mínimos.

En canto a funcionalidade base do prototipo existirá un módulo mestre e entre 1 e 255 escravos. Estes módulos escravos estarán provistos dun dispositivo Arduino e un módulo de comunicacións vía radio que aportan o necesario para facer medicións tanto analóxicas como dixitais do que se necesite, incluíndo o estado da súa propia fonte de enerxía, e o envío desta información cara un módulo mestre que será o encargado de tratala. Preténdese que estes módulos estean nun modo de aforro de enerxía e que se activen baixo o mandato do módulo mestre, de forma que podan estar facendo medicións en continuo e almacenándoas ata que o módulo mestre lle pida que llas envíe, ou estar en modo inactivo e facer a medición cando o mestre o requira.

En canto ao módulo mestre, estará composto por unha plataforma Arduino de maior envergadura ca a dos escravos, o seu correspondente módulo de comunicación vía radio e un módulo de comunicación IP para a conectividade cun PC. Este módulo mestre será o encargado de circular as ordes e a información recibida entre unha páxina HTML e os módulos escravos. Deste xeito dende a páxina poderán lanzarse ordes sobre os escravos e ver toda a información que aportan.

3 Antecedentes

O prototipo a deseñar que sae da análise e estudo do presente documento constará de catro partes fundamentais:

- **Plataforma de programación:** (Arduino) sobre a que se programarán os algoritmos de funcionamento.
- **Módulo de comunicación por radio:** que actuará como emisor e receptor entre os diferentes módulos que compoñen a rede.
- **Módulo de comunicación Ethernet:** que comunicará o módulo mestre cunha páxina HTML para poder ser consultada dende calquera dispositivo na mesma rede ethernet.
- **Sensores:** que realizarán as distintas medicións que se pretende que realicen os módulos escravos.
- **Actuadores:** que actuarán en base aos algoritmos de programación.

3.1. Plataforma de programación

A ferramenta utilizada na programación será Arduino, xa que así se especifica nos requisitos de deseño. Arduino é unha plataforma en código aberto e de hardware libre baseada nunha placa cun micro controlador e un entorno de desenrolo fácil.

A través das súas entradas analóxicas e dixitais pode facer medicións do seu entorno, as cales pode converter en saídas e controlar motores, LEDs, altofalantes ou calquera outro dispositivo que funcione con lóxica TTL.

A versatilidade en canto a equipamento que presenta esta marca permite un axuste concreto do produto as necesidades, podendo así elixir a placa que necesitamos en función do seu tamaño, procesador ou pines integrados.

Para o caso que nos ocupa, esta plataforma cumpre coas necesidades de deseño:

- **Para os escravos:** entradas analóxicas e dixitais, para ler a información dos sensores; fonte de alimentación a 5V ou 3.3V, para alimentalos; porto serie, para enviar e recibir información; interface para dispositivos I^2C e saídas dixitais con funcionalidade PWM para controlar actuadores.

- **Para o mestre:** máis de un porto serie para poder enviar a recibir información vía radio, e ao mesmo tempo comunicarse co PC e con outras extensións de comunicación; fonte de alimentación para os módulos exteriores.

A linguaxe de programación utilizada por esta plataforma está baseada na linguaxe *Processing* e é facilmente entendible e interpretable por usuarios non expertos. O software propio da marca permite o uso desta linguaxe con grandes prestacións.

Dado qué é unha plataforma de código aberto existen multitude de bibliotecas que nos axilizan e simplifican o traballo á hora de utilizar algúns sensores ou actuadores. Ademais permite unha compatibilidade case total con calquera tipo de sensor que aporte sinais entre 0 e 5 voltios e o mesmo para os elementos de saída.

A memoria integrada, a parte de conter as instrucións de arranque e o enlace para a carga de programas, permite gardar un programa para que a placa poda funcionar de forma autónoma sen estar conectada a un PC.

3.2. Módulos de comunicación por radio

A comunicación entre os escravos e o mestre farase sen fíos, dado que a idea fundamental é que os dispositivos escravos estean funcionando de forma autónoma afastados do mestre, e envíen a información vía radio cara este. Para iso cóntase con tres tecnoloxías que compren con esta función e ademais son compatibles con Arduino: Wi-Fi, GSM e ZigBee.

(a) Wi-Fi (b) GSM (c) ZigBee

Figura 3.2.0.1 – Tecnoloxías de comunicación por radio

3.2.1. Wi-Fi

O wifi é un mecanismo de conexión de dispositivos a través dun punto de rede común sen fíos. Traballa na banda de 2,4 GHz e ten un alcance aproximado de 20 metros. Na actualidade estase a traballar con redes wifi de 5 GHz por ser un espectro máis limpo de interferencias, pero o alcance é menor. Esta tecnoloxía permite un cifrado da información para aumentar a seguridade proporcionando uns protocolos propios integrados nos dispositivos como son o WEP (cifra os datos na súa rede para que solo o destinatario sexa capaz de descifralo), WPA (xeración dinámica da clave de acceso), IPSEC (pasarelas IP con autenticación e autorización de usuario), ou o filtrado de MAC (para escoller solo aquelas MAC que queremos que accedan).

A gran desvantaxe desta tecnoloxía e o baixo alcance e a cantidade de dispositivos que existen traballando na mesma banda de frecuencia, o que leva a unha maior cantidade de interferencias e unha consecuente redución da velocidade de transmisión.

Funcionamento:

O adaptador inalámbrico dun ordenador traduce os datos binarios a unha sinal de radio frecuencia de pulsos que se transmite a través dunha antena. Esta chega a un router inalámbrico que a decodifica (convertea en binario de novo). A continuación envía esta información a Internet usando unha conexión física por cable. O proceso é bidireccional, co router recibindo a información de internet e traducíndoa a unha sinal de radio que se envía por unha antena. Para que os datos enviados e recibidos polos dispositivos dunha rede non se mesturen cos doutra utilízase o SSID (Service Set Identifier) que é un código incluído nos paquetes de datos enviados, así os paquetes unicamente circularán entre os dispositivos co mesmo SSID.

3.2.2. GSM

O sistema global de comunicacións móbiles é o utilizado pola telefonía móbil para a transmisión de datos. Segundo a zona xeográfica na que se utilice, pode traballar en distintas bandas de frecuencia. En Europa traballa en torno a banda de 900 MHz.

O funcionamento é o seguinte:
Nunha rede GSM existe unha estación base é unha estación móbil.

- **Estación móbil:** está formada por unha tarxeta SIM que identifica ó usuario e por un dispositivo que a alberga e que ten un número de identificación chamado IMEI. Desta forma usuarios e dispositivos están identificados en todo momento pola rede.
- **Estación base:** dado que o sistema ten que ser capaz de soportar unha carga enorme de usuarios simultaneamente, a solución dunha única antena para todos sería inviable, xa que o espazo radio eléctrico se saturaría inmediatamente. É por iso polo que se procede a unha división das franxas de frecuencia utilizables entre diversas antenas no mesmo territorio, facendo así que dependendo do lugar xeográfico no que nos encontremos nos comuniquemos a través dunha banda ou doutra. Cada unha destas antenas forma unha estación base.

Deste xeito as antenas contiguas utilizan bandas de frecuencia distintas pero as que non están no mesmo radio repiten as bandas. Debido a esta condición aparece o controlador de estación base, un método que permite o transvasamento entre antenas se a potencia da sinal e maior noutra antena que nas que se está a utilizar nese momento.

3.2.3. ZigBee

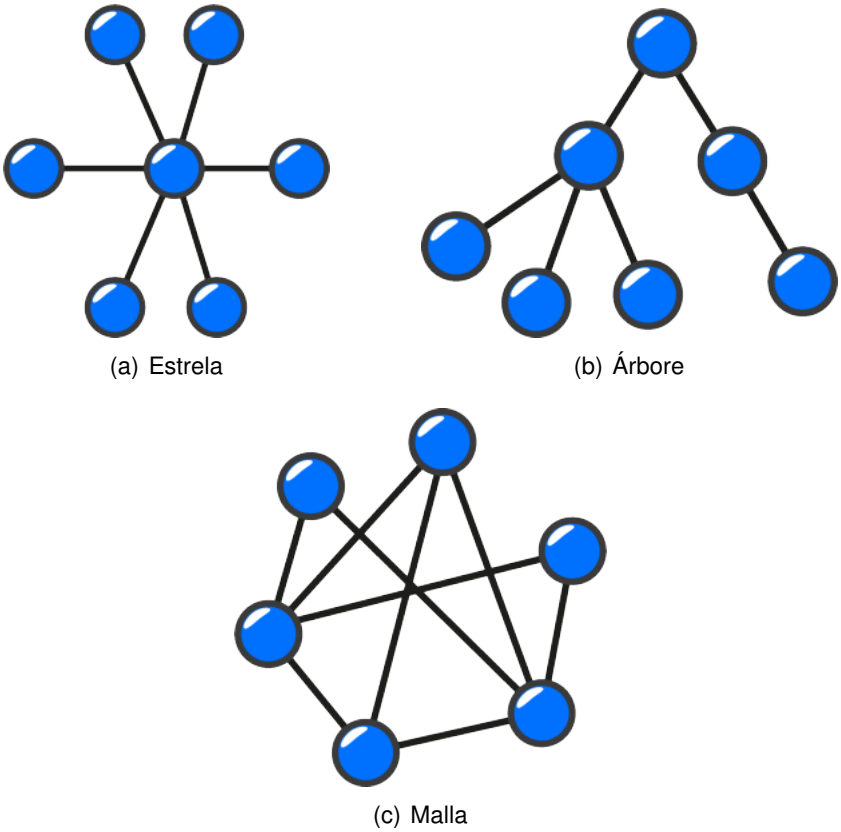
É un conxunto de protocolos de alto nivel para comunicacións sen fíos na radiodifusión dixital de baixo consumo. O seu uso está centrado nas aplicacións que requiren comunicacións seguras, con baixa taxa de envío de información e consumo moi reducido. Este método traballa na banda ISM (uso industrial, científico e médico) de 868 MHz en Europa, e en xeral en 2,4 GHz. Aínda que o seu consumo sexa baixo, o seu alcance é bastante alto, chegando ao quilómetro de distancia.

Existen tres tipos de dispositivos ZigBee:

- **Coordinador:** imprescindible un por rede para controlala e indicar os camiños que deben seguir os dispositivos para conectarse entre eles.
- **Router:** actúa como enlace entre dispositivos separados na topoloxía da rede.
- **Dispositivo final:** ten a funcionalidade necesaria para comunicarse co seu coordinador, e as súas calidades son limitadas. A maioría do tempo estará a espera de ordes nun estado de *stand-by*.

A topoloxía de rede entre estes dispositivos pode ser:

- **En estrela:** o coordinador está no centro da rede e os escravos conéctanse independentemente a el.
- **En árbore:** o coordinador está na base da rede e del van saíndo as diferentes sub redes.
- **En malla:** os nodos están interconectados entre si, proporcionando vías alternativas en caso de que un elemento da cadea falle.



(a) Estrela (b) Árbore

(c) Malla

Figura 3.2.3.1 – Topoloxías do sistema ZigBee

O funcionamento do estándar ZigBee é igual ca o de calquera sistema de radio: unha información binaria codifícase en pulsos, que se envían a través dunha antena. Do outro lado outra antena recibe eses pulsos que decodificará e traducirá de novo a binario. O módulo

ZigBee comunicarase a través de dous terminais cos dispositivos finais: un para recibir e outro para transmitir, é dicir, traballa co protocolo das comunicacións en serie.

Os paquetes de datos que manexa ZigBee están formados por tramas de 104 bytes, que están numeradas para asegurarse de que o contido se recibiu sen erros. Entre as estruturas importantes están o *ACK* ou acuse de recibo, que funciona como realimentación entre receptor e emisor informando de que se recibiu un paquete; o paquete *MAC*, que identifica o nodo que recibirá o mensaxe; e o paquete baliza, que se encarga de facer a función espertador para os nodos que están nun modo de baixo consumo denominado *SLEEP*, que consiste en que o módulo estea inactivo ata que recibe unha sinal para espertarse e transmitir e recibir datos.

3.3. Módulo de comunicación Ethernet

Coa finalidade de crear un deseño universal e facilmente adaptable a calquera instalación, a visualización de datos e o control dos dispositivos farase dende un entorno facilmente asimilable e manexable por calquera persoa. Para iso crearase unha páxina web en HTML a cal accederá aos datos e aos dispositivos mediante o protocolo de Ethernet.

Este estándar baséase nas redes de área local (LAN) e ten un formato predefinido que contén os seguintes campos:

- **Preámbulo:** indica o inicio da trama e forza o dispositivo a sincronizarse.
- **Delimitador de inicio:** indica que o paquete comeza nese momento.
- **MAC:** indica as direccións físicas de orixe e de destino dos dispositivos.
- **Ethernetype:** indica o protocolo ao que poden estar sometidos os datos que contén o payload.
- **Payload:** lugar onde se aloxan todos os datos que se van transmitir.
- **CRC:** Control de Redundancia Cíclica. É un campo de 4 bytes que contén un valor de verificación para saber se os datos se transmitiron correctamente.
- **Gap final:** 12 bytes baleiros para asegurarse unha separación suficiente entre tramas.

Existen multitude de tecnoloxías dentro do estándar de ethernet, que varían en función do medio físico empregado e da topoloxía da rede, facendo variar a velocidade de transmisión.

Os parámetros básicos a considerar nunha conexión Ethernet son a MAC, a IP e a porta de enlace ou gateway.

- **MAC:** identifica o dispositivo físico que soportará a rede.
- **IP:** etiqueta numérica que identifica un dispositivo de forma lóxica e xerárquica.
- **Porta de enlace:** identificador da dirección de conexión cara Ehternet.

3.4. Sensores e actuadores

Dado que a finalidade última do deseño proposto nesta documentación é a transmisión de medidas de variables físicas do entorno, dalgún xeito haberá que obtelas. As medicións fanse co que se denomina sensores, que son dispositivos capaces de detectar magnitudes físicas ou químicas e transformalas en variables eléctricas analóxicas ou dixitais. Como parámetro crítico á hora de elixir os sensores a utilizar é que a súa resposta ten que variar no rango de 0 a 5 voltios, no caso das analóxicas ou transmitir en niveis TTL no caso das dixitais. En caso de que non cumpran con esta especificación deberanse acompañar dun circuíto de acondicionamento.

Para a lectura dos sensores a gama arduino conta con entradas analóxicas e dixitais, cuxo número varía dependendo da placa empregada. No caso das entradas analóxicas, para ser cuantificadas, fanse pasar por un conversor A/D de 10 bits, que ten como tensión de referencia interna 5 voltios, a cal pode ser modificada por unha menor. Con el o que se consigue é cuantificar a variación de 0 a 5 voltios con números entre o 0 e 1024.

No caso das dixitais, pasan por un buffer que detecta o nivel de tensión que traen; en caso de superar un limiar consideraranse de nivel e alto e no caso contrario de nivel baixo.

Por outro lado temos os actuadores, que son os elementos de saída que se conectan á placa e que se activarán ou desactivarán en función dos algoritmos de cálculo que se implementen no procesador. Estes actuadores son básicamente leds, motores ou outras placas que realizan funcións de medida ou actuación. De forma similar aos sensores, tamén poden actuar de forma dixital ou analóxica, aínda que as saídas da placa arduino únicamente son dixitais.

No caso de utilizarse como saídas dixitais funcionarán no modo *todo ou nada*, é dicir, que ou reciben 5 voltios ou cero. Se a intención é que funcionen con tensión reguladas entre 0 e 5 habería que utilizar as saídas PWM, que ainda que son dixitais realizan un control por ancho de pulso que regula a tensión media entregada a unha carga en función do ciclo de traballo dun sinal. Estas saídas PWM teñen un rango de traballo configurable en 8 bits (0-255).

4 Normas e referencias

4.1. Disposicións legais e normas aplicadas

IEEE 802.15.4 Redes sen fíos de área persoal. Estándar utilizado por ZigBee Alliance

IEEE 802.3 / **ISO 8802.3** Normativa sobre sistemas TCP/ IP e ethernet

ISO 8859 Liguanxe HTML e código ASCII

ISO / **IEC 15445** Linguaxe HTML

4.2. Bibliografía

[1] BLUM, J.; *Arduino a fondo*, 1ª ed, Madrid, Anaya Multimedia, (2014).

[2] ARTERO, O.; *Arduino: curso práctico de formación*, 1ª ed, San Fernando de Henares, RC Libros, (2013).

[3] FALUDI, R.; *Building wireless sensor network*, 1ª ed, USA, O'Reilly Media, (2011).

[4] IGOE, T.; *Making things talk*, 2ª ed, England, Maker Media Inc, (2011).

[5] MARGOLIS, M.; *Arduino cookbook*, 2ª ed, O'Reilly, (2011).

[6] SUÁREZ PEÑARANDA, V., *Edición de proyectos de fin de grado con LaTeX*, (2014).

4.3. Programas de cálculo e software

TeXnicCenter Editor de textos en formato LaTex para Windows.

Arduino IDE 1.6.4 Entorno de desenrolo e programación das plataformas Arduino.

Cadence OrCAD 10.5 Entorno de simulación de circuitos electrónicos.

AutoCAD 2015 Software de deseño asistido por computadora utilizado para debuxo 2D e modelado 3D.

X-CTU *XBee Configuration and Test Utility* Aplicación de configuración e proba para módulos de radio de Digi.

4.4. Outras referencias

[i] *Cómo funciona la tecnología WiFi*, Josienita Borlongan. Dispoñible en: http://www.ehowenespanol.com/funciona-tecnologia-wifi-como_10752/

[ii] *Guía rápida del funcionamiento de una red Wifi*. Dispoñible en: http://www.ordenadores-y-portatiles.com/red-wifi.html

[iii] *Como Funciona Una Red Inalambrica WIFI*. Dispoñible en: http://www.taringa.net/posts/info/15488304/Como-Funciona-Una-Red-Inalambrica-WIFI.html

[iv] *Cómo funciona la red GSM* , Pablo Alejandro Fain, [30 de maio de 2009]. Dispoñible en: https://www.pablofain.com/como-funciona-la-red-gsm/

[v] *Todo sobre ZigBee, la tecnología ultrabarata para comunicación inalámbrica*, Manuel J. Gutiérrez, [10 de agosto de 2015]. Dispoñible en: http://www.elandroidelibre.com/2015/08/todo-sobre-zigbee-la-tecnologia-ultrabarata-para-comunicacion-inalambrica.html

[vi] *Comenzando con ZigBee.* , [14 de febreiro de 2012]. Dispoñible en: http://webdelcire.com/wordpress/archives/1714

[vii] *Crear servidor web con Arduino*, [24 de xaneiro de 2014]. Dispoñible en: http://diymakers.es/crear-servidor-web-con-arduino/

[viii] *Conoce tu router (I): dirección IP, máscara de subred y puerta de enlace*, Emmanuel Jiménez, [4 de novembro de 2010]. Dispoñible en: http://www.xatakaon.com/equipos-de-red/conoce-tu-router-i-direccion-ip-mascara-de-subred-y-puerta-de-enlace

[ix] *Aprendiendo Arduino*. Dispoñible en: https://aprendiendoarduino.wordpress.com/2015/03/29/memoria-flash-sram-y-eeprom/

[x] *Clock speed*, Vangie Beal. Dispoñible en: http://www.webopedia.com/TERM/C/clock_speed.html/

[xi] *Chosing a MCU for your next design; 8 bit or 32 bit?*, Ingar Fredriksen, MCU marketing director, Atmel and Pal Kastnes, MCU Applications Staff Engineer, Atmel. Dispoñible en: http://www.atmel.com/images/45107a-choosing-a-mcu-fredriksen_article_103114.pdf

[xii] *Arduino Ethernet Shield - Controla tu casa por internet*, [16 de novembro de 2014]. Dispoñible en: http://www.educachip.com/arduino-ethernet-shield/#

[xiii] *HTML(5) Tutorial*, W3Schools. Dispoñible en: http://www.w3schools.com/html/

[xiv] *Wikipedia: the free encyclopedia*. Wiki en Internet. Wikimedia Foundation, Inc. 2001. Dispoñible en: http://en.wikipedia.org/

[xv] *Arduino*. Páxina de referencia da tecnoloxía Arduino. Arduino, Inc. 2016. Dispoñible en: http://www.arduino.cc/

[xvi] *Atmel*. Fabricante de procesadores. Atmel Corporation 2015. Dispoñible en: http://www.atmel.com/

[xvii] *DIGI*. Fabricante de módulos de radio frecuencia. Digi International, Inc. 1996-2016. Dispoñible en: http://www.digi.com/

[xviii] *Plantilla para traballos de fin de grao en Latex*. Vicente Suárez Peñaranda, 2015. Dispoñible en: http://lucas.cdf.udc.es/nodos/vercont.php?tipo=3&pos=80&ent=bccratibinum000

5 Definicións e abreviaturas

IP *(Internet Protocol)*: Protocolo de internet. É un estándar que se emprega no envío e recepción de información mediante unha rede que reúne paquetes conmutados.

PC *(Personal Computer)*: Ordenador persoal.

HTML *(HyperText Marks Language)*: Linguaxe de marcas de híper texto. É un estándar de creación de páxinas web. Define unha estrutura básica e un código para a definición de contidos.

LED *(Light Emmitting Diode)*: Diodo emisor de luz. Tipo de semiconductor sólido que ao recibir unha pequena corrente emite luz grazas as súas propiedades fotoluminiscentes.

TTL *(Transistor Transistor Logic)*: Lóxica de transistor a transistor. Tecnoloxía de construción de circuítos dixitais electrónicos baseada no uso de transistores como elementos de entrada e saída.

I^2C ***(Inter Integrated Circuit)*:** Circuítos inter integrados. É un bus de comunicacións en serie que utiliza dous sinais: unha liña de pulsos de sincronización e un liña de datos.

TWI *(Two Wire Interface)*: Interface de dous fíos. É o mesmo sistema que o anterior.

V *(Voltios)*: Unidade de medida do sistema internacional para a medida de potencial eléctrico.

PWM *(Pulse Widht Modulation)*: Modulación por ancho de pulso. Consiste na transformación dunha sinal continua nunha onda cadrada que oscile entre un valor nulo e o máximo a unha frecuencia determinada para variar a tensión media entregada as cargas.

Wi-Fi *(Wireless Fidelity)*: Fidelidade inalámbrica. Mecanismo de conexión de forma inalámbrica.

Hz *(Hertzio)*: Unidade de frecuencia do Sistema Internacional de Unidades.

MHz *(Mega hertzio)*: 10^6 hertzios.

GHz *(Xiga hertzio)*: 10^9 hertzios.

WEP *(Wired Equivalent Privacy)*: Privacidade equivalente a cableado. Sistema de cifrado para redes sen fíos.

WPA *(Wi-Fi Protected Access)*: Acceso a Wi-Fi protexido. Sistema de protección de redes inalámbricas.

IPSEC *(Internet Protocol SECurity)*: Seguridade de protocolo de internet. Conxunto de protocolos cuxa función e asegurar as comunicacións sobre o IP.

MAC *(Media Address Control)*: Control de acceso ao medio. É un identificador de 48 bits que corresponde única e inequivocamente a unha tarxeta ou dispositivo de rede.

GSM *(Global System for Mobile communications)*: Sistema global para as comunicacións móbiles. Sistema de comunicación de telefonía móbil dixital.

SIM *(Subscriber Identity Module)*: Módulo de identificación de aboado. Tarxeta intelixente utilizada en móbiles.

IMEI *(International Mobile System Equipment Identity)*: Sistema internacional para a identificación de equipos móbiles. Código de identificación de aparellos que funcionan baixo o estándar GSM.

ISM *(Industrial, Scientific and Medical)*: Industrial, científica e médica. Bandas de frecuencia electro magnética reservadas internacionalmente para usos non comerciais nas áreas industrial, científica e médica.

LAN *(Local Area Network)*: Rede de área local. Rede de computadoras que abarca unha área reducida.

A/D *(Analog/Digital)*: Analóxico dixital.

kB *(kilo bytes)*: Medida de memoria equivalente a 1024 bits.

SRAM *(Static Random Access Memory)*: Memoria estática de acceso aleatorio. Tecnoloxía baseada en semicondutores capaces de manter datos mentres siga alimentada sen necesidade de refresco.

EEPROM *(Electrically Erasable Programmable Read Only Memory)*: Memoria de solo lectura programable e borrable electricamente. É un tipo de memoria que pode ser modificada de xeito eléctrico sen intervención de raios ultravioleta.

USB *(Universal Serial Bus)*: Bus serie universal. Bus estándar que define cables, conectores e protocolos de comunicacións eléctricas en serie.

SPI *(Serial Peripheral Interface)*: Interface de periféricos serie. Estándar de comunicacións usado na transferencia de información entre circuítos integrados en equipos eléctronicos.

UART *(Universal Asynchronous Receiver Transmitter)*: Receptor - transmisor asíncrono universal. Dispositivo que controla os portos e dispositivos serie.

API *(Aplication Programming Interface)*: Interface de programación de aplicacións. Conxunto de subrutinas, funcións e procedementos que ofrecen certas bibliotecas de funcións.

CR *(Carriage Return)*: Retorno de carro. Salto de liña en ASCII.

DHCP *(Dynamic Host Configuration Protocol)*: Protocolo de configuración dinámica de host. Protocolo de rede de tipo cliente servidor.

6 Requisitos de deseño

Os requisitos mínimos de deseño son os seguintes:

- Basear o deseño na plataforma Arduino.
- Comunicación vía RF (ZigBee, WiFi ou GSM).
- Contar con unha unidade mestra que estableza a topoloxía de rede.
- Utilizar nodos repetidores en caso de problemas na comunicación.
- Soportar 255 esclavos.
- Soportar medidas analóxicas e dixitais.
- Monitorización da fonte de enerxía.
- Xeración de alarmas de estado.

Partindo desta referencia, poderán ser engadidas novas funcionalidades, sempre e cando non entren en conflito coas existentes e aporten valor ao deseño final.

7 Análise das solucións

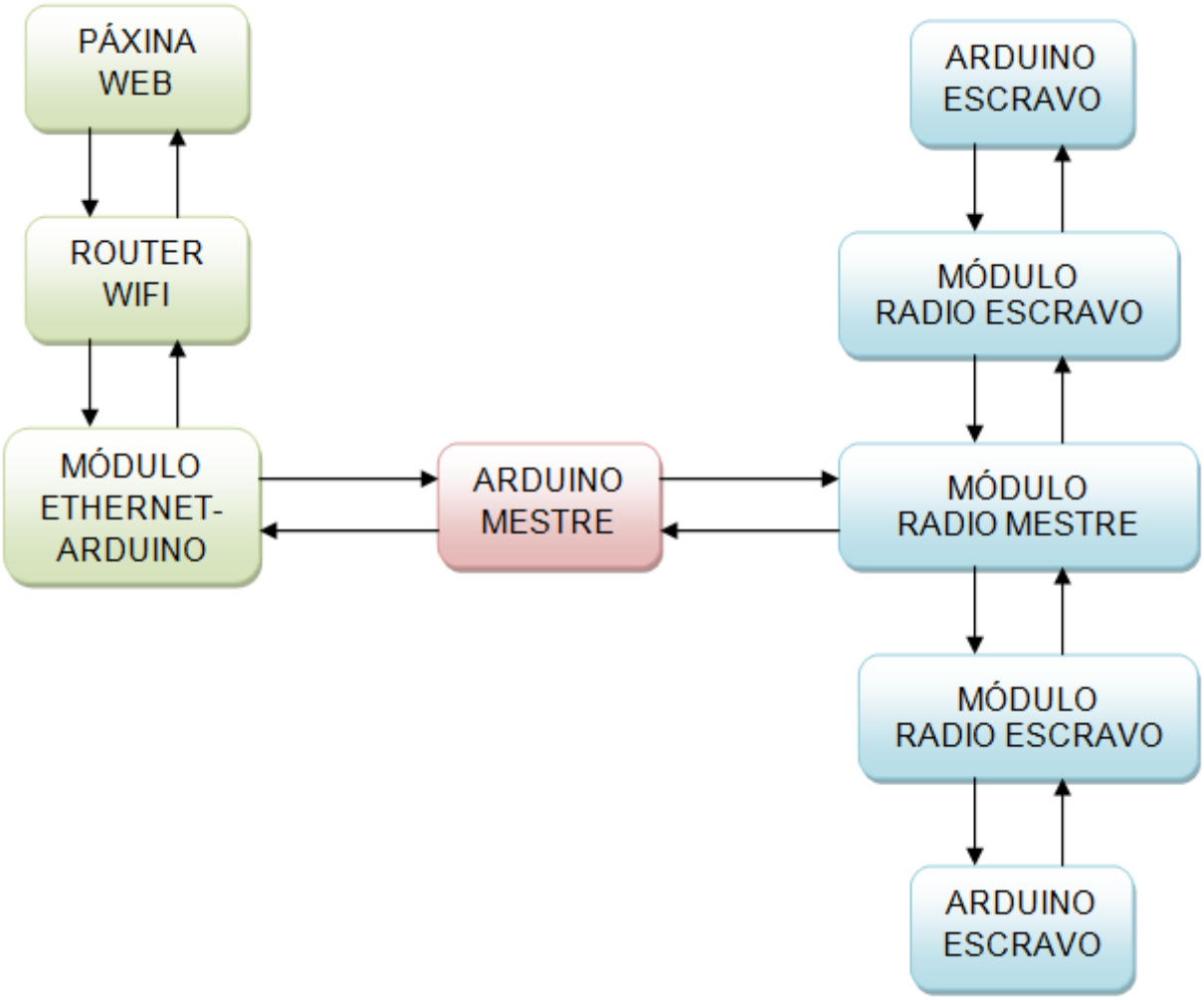


Figura 7.0.0.1 – Esquema de funcionamento

O fundamento práctico deste deseño é o seguinte:

Por un lado temos a parte de comunicación de internet/ethernet, que estará encabezada por unha páxina web. Dende ela farase o control dos dispositivos remotos, a través de botóns e elementos de entrada; así como tamén servirá para a visualización dos datos proporcionados polos mesmos. A comunicación entre o Arduino e o servidor web faise a través dun módulo de ethernet e un router (neste caso un router wifi). Deste xeito a páxina web creada terá unha dirección IP á que será posible acceder dende calquera dispositivo conectado na mesma rede

local. Todo isto controlado dende unha plataforma Arduino que actuará como procesadora tanto das peticións recibidas dende a páxina como dos datos recibidos dende os módulos escravos.

Por outro lado temos a parte de comunicación vía radio. A plataforma mestra recibirá as peticións da páxina web, que basicamente serán solicitudes para ver as medicións realizadas e o estado da fonte de enerxía dos módulos remotos, e enviaralles a estes, vía radio unha petición de información. No caso de que os módulos remotos estean a unha distancia axeitada do mestre recibirán no seu módulo de radio a petición e a placa á que están conectados actuará en consecuencia. Se pola contra a plataforma mestra envía unha petición de información aos escravos e algún non responde, acudirase a un módulo intermedio que actúe como repetidor, xa que os módulos escravos unicamente son capaces de comunicarse co mestre ou cun repetidor e non entre eles.

7.1. Plataforma de programación

Dado que un dos requisitos do deseño é a utilización da placa de desenrolo Arduino, o que queda é escoller o modelo desta.

Na actualidade existen 10 modelos de placas que se poderían utilizar:

- Arduino UNO
- Arduino 101
- Arduino PRO
- Arduino PRO Mini
- Arduino Micro
- Arduino Nano
- Arduino Mega
- Arduino Zero
- Arduino Due

O criterio fundamental á hora de escoller a correcta é o do uso que vai recibir, así pois, non será a mesma a placa que actúe como mestra da rede que a dos que actúen como escravos.

7.1.1. Arduino escravo

Comezando polos escravos, a función destes será basicamente a de tomar medidas cando se requiran e envialas a través do módulo de radio, por tanto non se necesita unha placa moi

potente para que cumpran a súa función. Neste caso, e con vistas a crear módulos completamente independentes para instalar en exteriores, buscaranse os de menor tamaño e custo. Así a variedade a escoller limítase a Arduino PRO, PRO mini, Micro e Nano.

A continuación mostrase unha comparativa destes modelos:

	PRO	**PRO mini**	**Micro**	**Nano**
Microcontrolador	ATmega168/ ATmega328*	ATmega328	ATmega32U4	ATmega168/ ATmega328*
Voltaxe operativo (V)	3,3/5*	3,3/5	5	5
Pines dixitais (PWM)	14(6)	14(6)	20(7)	14(6)
Entradas analóxicas	6	6	12	8
Memoria flash (kB)	16/32*	32	32	16/32*
SRAM (kB)	1/2*	2	2,5	1/2*
EEPROM	512 bytes/ 1 kB*	1 kB	1 kB	512 bytes/ 1kB*
Frecuencia de traballo (MHz)	8/16	8/16	16	16
Conexión USB	NON	NON	SI	SI
Tamaño (mm)	52x53	18x33	48x18	45x18

* Existen dous modelos distintos en base ao procesador empregado (1ª cifra ATmega168, 2ª cifra ATmega328)

Taboa 7.1.1.1 – Comparativa de modelos de Arduino escravo

En primeiro lugar explicar as características que se comparan aquí:

Microcontrolador: a gama Arduino (e similares) funciona con procesadores da marca ATMEL. Este é o cerebro da placa e será o que realice da forma máis axeitada o procesado dos datos e as actuacións que se requiran na programación da placa.

Voltaxe operativo: tensión de funcionamento da placa e límite admisible de tensión nos pines de entrada.

Pines dixitais: son aquelas entradas ou saídas que funcionan con dous niveis lóxicos: nivel alto ou nivel baixo. Algúns destes pines teñen a función PWM (Modulación por Ancho de Pulso), que o que fai é variar segundo un parámetro denominado ciclo de traballo o tempo que as saídas se encontran a nivel alto, conseguindo así variar a tensión media entregada as cargas.

Entradas analóxicas: contactos metálicos conectados a un convertedor A/D de 10 bits que traduce unha tensión de entrada entre cero e o voltaxe de referencia a un número entre 0 e 1024.

Memoria flash: espazo onde se almacena o programa que se carga na placa. Nela inclúese o programa de arranque.

SRAM: memoria de acceso aleatorio que sirve como almacenamento das variables coas que traballa o programa de Arduino. Esta memoria borrarase cada vez que non está alimentada a placa.

EEPROM: memoria destinada a programadores para almacenar información a longo prazo.

Frecuencia de traballo: velocidade á cal o microcontrolador executa instrucións.

Conexión USB: conexión física entre a placa Arduino e un PC (ou similar), para a programación do mesmo.

A conexión física dun porto USB para a programación (e nalgún caso da alimentación) é un factor importante á hora de escoller a placa a utilizar. No caso das dúas primeiras opcións esta característica non está incluída, polo que se fai necesario a incorporación dun elemento externo para poder programar o micro controlador. É por iso polo que se desbotan estas dúas primeiras opcións.

En canto a escoller entre o modelo Micro e o modelo Nano, calquera dos dous podería ser empregado sen discriminación. A súa diferenza principal radica no procesador integrado. Neste caso escollerase o modelo Nano, que aínda que é menos potente que o Micro é máis económico e cumpre sobradamente coas especificacións técnicas requiridas.

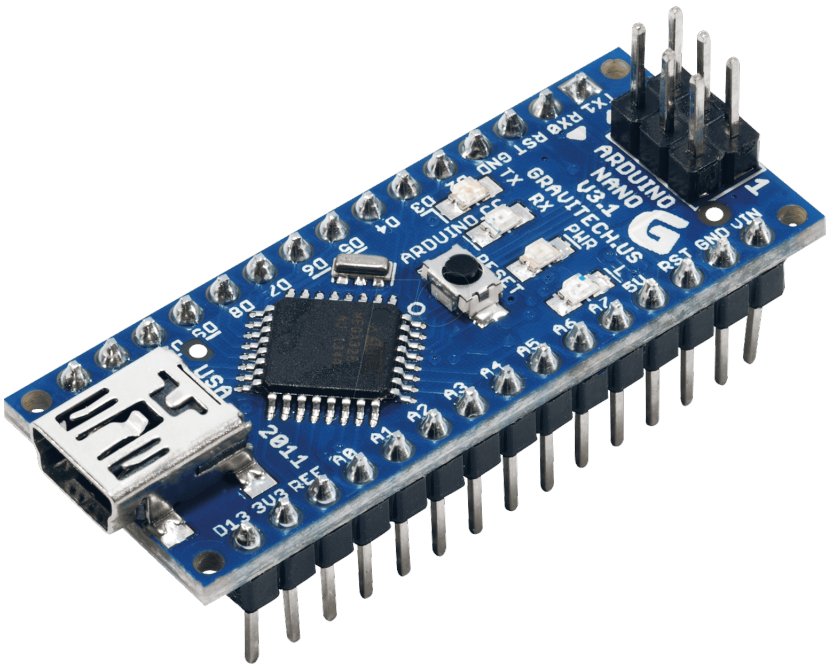


Figura 7.1.1.1 – Arduino NANO

Características:

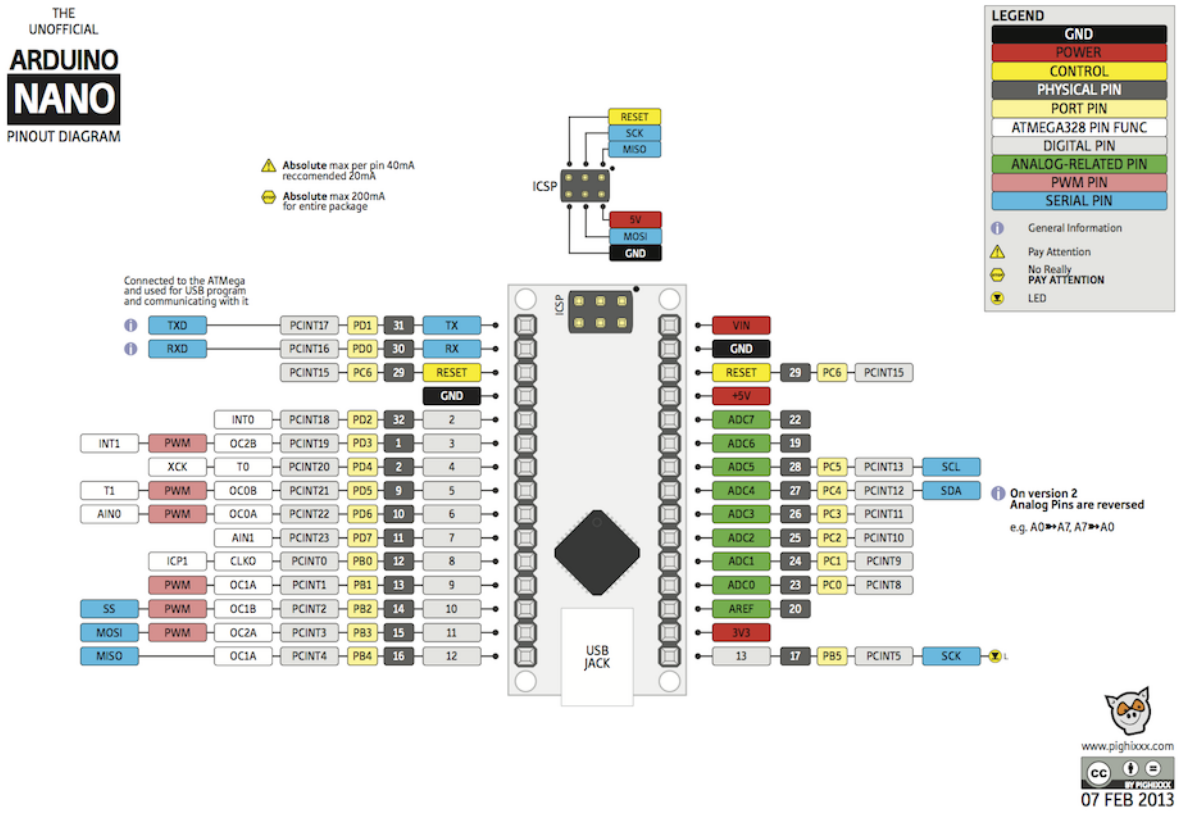


Figura 7.1.1.2 – Pinout Arduino NANO

- **Potencia:**

  O Arduino Nano pode ser alimentado a través de tres fontes distintas:

  - **VIN:** entrada para fontes de tensión externa sen regular entre 6 e 20 voltios.
  - **5V:** entrada para unha fonte de tensión regulada de 5 voltios.
  - **Mini B USB:** conexión USB integrada na placa.

  A fonte de alimentación seleccionase de xeito automático entre a que proporcione maior nivel. Tamén é capaz de proporcionar alimentación a elementos externos a través dos contactos 5V e 3V3 (ademais do terminal de terra).

- **Entradas e saídas:**

  - *Dixitais*

    Os 14 contactos designados como pines dixitais poden usarse tanto como entrada como saída. Funcionan a 5 voltios e son capaces de entregar ou recibir ata 40 mA. Como protección interna contan cunha resistencia de pull-up de entre 20 e 50 kOhms. Algúns destes contactos teñen asignadas funcións especiais:

    - **Serie:** RX0 e TX1. Usadas na recepción e transmisión de datos en serie en formato TTL.

- **Interrupcións externas:** D2 e D3. Contactos configurables para activar unha interrupción do procesador a nivel baixo, nun flanco de subida ou de baixada ou nun cambio de nivel.
- **PWM:** D3, D5, D6, D9, D10 e D11. Saídas configurables para unha modulación de ancho de pulso de 8 bits.
- **SPI:** D10 (SS), D11 (MOSI), D12 (MISO) e D13 (SCK). Contactos empregados na comunicación serie entre dispositivos baixo o estándar SPI.
- **LED:** D13. No contacto do pin 13 hai un LED incrustado na placa que funciona do mesmo xeito que o contacto asociado.

• *Analóxicas*

As 8 entradas analóxicas da placa fan medicións entre os 0 e os 5 voltios a través do convertedor A/D de 10 bits de resolución integrado. O marxe superior do rango de medida pode ser variado a través dunha función interna ou dunha fonte de tensión colocada no pin AREF. É importante ter en conta que este marxe unicamente pode ser variado entre os 0 e os 5 voltios e que debe indicárselle mediante a función adecuada á placa de Arduino, do contrario as fontes de referencia internas poranse en curtocircuíto e a placa danarase. Os contactos analóxicos con función especializada son o A4 e o A5 que fan a función SDA e SCL para as comunicacións baixo o estándar I^2C (usado por exemplo nas pantallas de visualización).

7.1.2. Arduino mestre

A elección da placa Arduino mestra é de maior complicación, xa que esta será a peza clave do deseño final. En torno a ela fluirá toda a información da rede e será quen se encargue de establecer os criterios finais da comunicación, por esta razón escollerase unha placa cun procesador máis potente ca o das placas escravas. Os Arduinos de nivel superior son: Arduino MEGA, Arduino ZERO e Arduino DUE, os cales compararemos en canto as súas características fundamentais:

	MEGA	**ZERO**	**DUE**
Microcontrolador	ATmega2560, 8 bits	ATSAMD21G18, 32 bits	AT91SAM3X8E, 32 bits
Voltaxe operativo (V)	5	3,3	3,3
Pines dixitais (PWM)	54(15)	20(18)	54(12)
Entradas analóxicas	16	6	12
Memoria flash (kB)	256	256	512
SRAM (kB)	8	32	96
EEPROM	4 kB	-	-
Frecuencia de traballo (MHz)	16	48	84
UART	4	2 (nativos e programables)	4

Taboa 7.1.2.1 – Comparativa de modelos de Arduino mestre

Neste caso o tamaño non é un factor crítico, polo que non se inclúe na táboa; así como a

conexión USB, que nos dispositivos avanzados aparece en todos. Como novidade na comparación aparece o elemento UART:

UART: porto de comunicación utilizado na recepción e envío de información serie asíncrona. Funciona baixo o estándar TTL traducindo información en pulsos de nivel alto e baixo que serán entendidos como datos binarios.

A principal diferenza entre as tres placas é o microcontrolador. Mentres que a MEGA ten un de 8 bits, as outras dúas integran un de 32 bits. Os procesadores con máis bits teñen maior potencia de cálculo, maior precisión e maior velocidade de proceso da información, xa que en un único ciclo máquina son capaces de tratar catros veces máis información. Non obstante, e segundo o fabricante (ATMEL) a fiabilidade dos de 8 bits é maior, ademais de que a gran maioría das aplicacións de usuario integrables nestas placas son cubertas con garantías con estes procesadores.

Outro punto clave na elección da placa é a dispoñibilidade dunha memoria EEPROM, onde se gardarán variables de forma permanente cando os circuítos non dispoñan de alimentación. A única da gama que dispón dela é a MEGA, aínda que as outras admiten módulos externos para incluírlla. Por último como factor crítico aparecen os UARTS, que son os portos de comunicación con outros dispositivos. É importante que conten con mais de un xa que a comunicación vía radio farase por un deles e sería interesante poder contar con máis en caso de querer comunicarse a través do PC coa placa ou en vistas de incorporarlle outro elemento que traballe co estándar de comunicacións TTL asíncronas.

Tras realizar unha comparación dos modelos que ofrece o mercado decídese que a placa escollida para actuar como mestra sexa a Arduino MEGA, que aínda que non ten o procesador máis rápido e potente da gama cumpre sobradamente coas especificacións requiridas. A condición de voltaxe operativo de 5 voltios implica unha comodidade engadida, xa que de conectárselle algunha extensión non se precisará de hardware externo que eleve a tensión dos contactos, que nas outras dúas é de 3,3 voltios. Conta coa memoria EEPROM, que permite almacenar variables cada vez que se desconecte da fonte de enerxía e sobre todo conta con varias unidades UART.

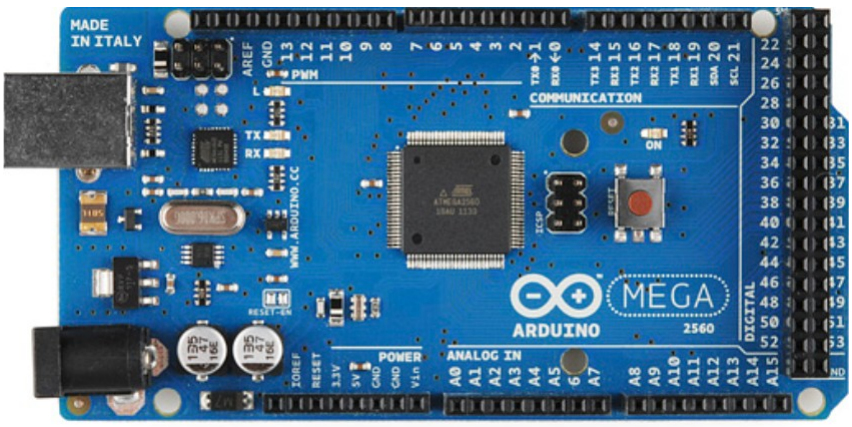


Figura 7.1.2.1 – Arduino MEGA

Características:

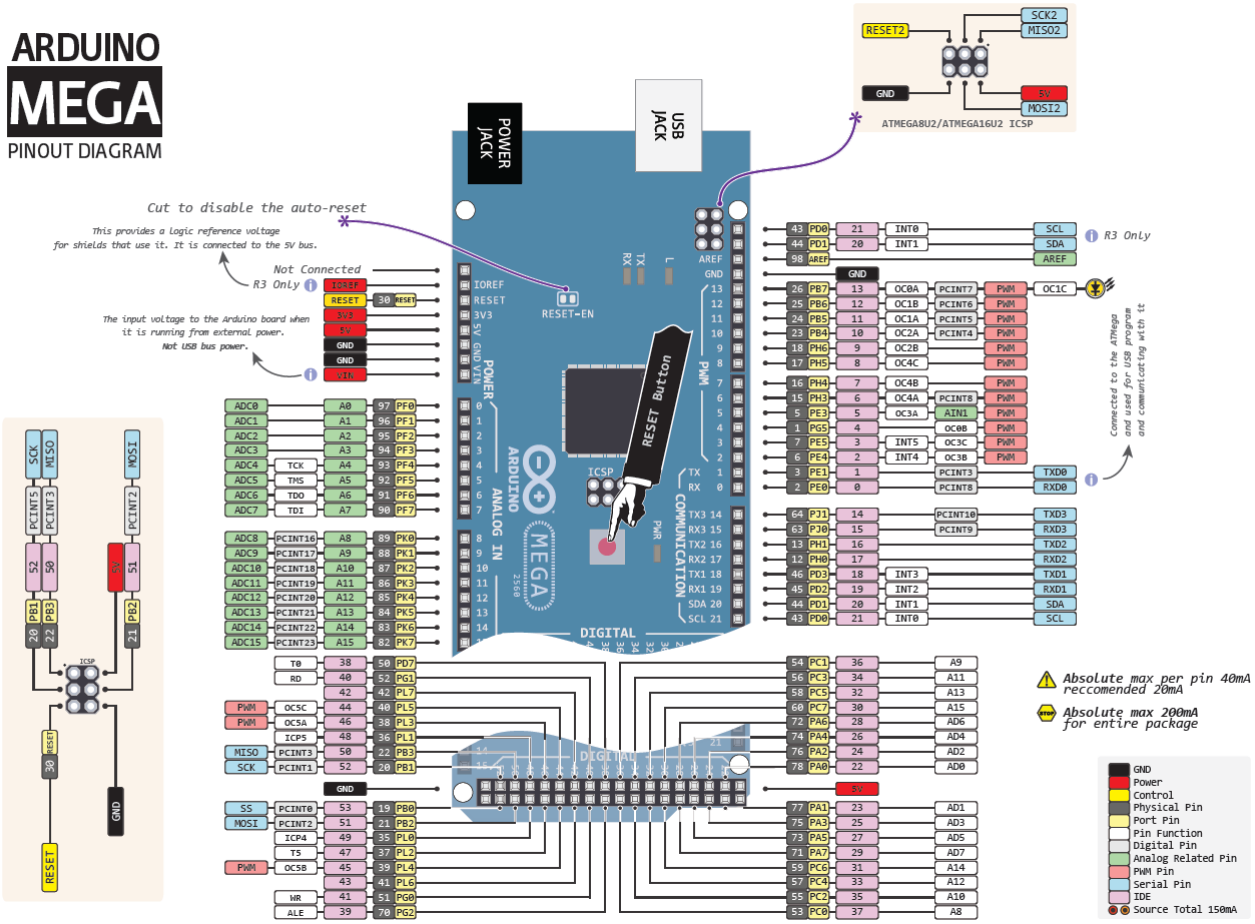


Figura 7.1.2.2 – Pinout Arduino MEGA

- **Potencia:**

  A alimentación é similar á da placa Nano, coa diferenza de que o pin VIN que ten como función alimentar a placa mediante unha fonte sen regular de entre 7 e 12 voltios, ademais deste contacto na placa ten asociado un conector Jack. Igual que nas demais placas da gama Arduino, existen dúas fontes de tensión: unha de 5 voltios e outra de 3,3V. Por último aparece un novo contacto, rotulado co nome IOREF, que serve para proporcionar unha referencia a aquelas extensións da placa que poden operar con 5 voltios ou con 3,3, indicándolles con cal delas están a traballar os contactos de entradas e saídas.

- **Entradas e saídas:**

  - *Dixitais:*

    Os 54 contactos dixitais da placa poden ser utilizados tanto como entrada como saída pero existen uns cantos con funcións específicas:

    - **Serie:** existen catro pares de contactos utilizados na transmisión de datos serie; o par 0(RX) e 1(TX) ademais incorporan a función de programación vía USB. Os

outros tres son: Serie1, 19(RX)-18(TX); Serie2, 17(RX)-16(TX); Serie3, 15(RX)-14(TX).

- **Interrupcións externas:** 2 (Interrupción 0), 3 (Interrupción 1), 18 (Interrupción 5), 19 (Interrupción 4), 20 (Interrupción 3) e 21 (Interrupción 2).
- **PWM:** 2 a 13 e 44 a 46. Proporcionan unha saída regulada TTL de 8 bits.
- **SPI (Serial Peripheral Interface):** 50 (MISO), 51 (MOSI), 52 (SCK) e 53 (SS), que están repetidos na cabeceira ICSP.
- **LED:** igual que na maioría de placas o pin 13 contén un LED asociado integrado na mesma.
- I^2C **(TWI):** 20 (SDA) e 21 (SCL). Utilizado no estándar de comunicación por dous cables.

- *Analóxicas*

  Estas 16 entradas analóxicas tamén están integradas nun convertedor A/D de 10 bits de resolución. A referencia interna de medida son 5 voltios que poden ser modificados de forma software, mediante codificacións internas; ou de forma hardware no contacto AREF.

7.1.3. Programación das placas Arduino

Arduino utiliza unha linguaxe de programación baseado nos estándares Processing e Wiring, aínda que pode ser tamén programado con linguaxes informáticos tales como C ou C++. Conta cun entorno de desenrolo de aplicacións propio denominado Arduino IDE, que utiliza os estándares antes mencionados e que ofrece todas as facilidades necesarias para a utilización destes dispositivos. Os códigos que se editan dende esta aplicación chámanse sketches. Dado que a finalidade deste traballo non é alcanzar un dominio total da programación desta plataforma explicaranse unicamente aquelas funcionalidades que se utilicen na programación dos módulos do deseño. A programación destas plataformas divídese en tres bloques principais: estruturas, variables e funcións.

- **Estruturas**

  - **Comúns**
    - **setup():** inicializa as variables, as configuración dos pines e todo o que necesite arrancar para ser empregado no programa principal. Execútase unha vez cando a placa recibe a alimentación.
    - **loop():** é o bucle principal de programación. Nesta función descríbese o funcionamento da placa e está executándose infinitamente ata que recibe unha instrución de parada ou a placa se queda sen alimentación.
  - **Estruturas de control**
    - **if:** é unha función condicional que avalía unha expresión de comparación e actúa en consecuencia si se cumpre.

- **if - (else if) - else:** é a extensión da función anterior pero contemplando outras actuacións en caso de que non se cumpran as condicións impostas tras cada if.
- **while:** comproba unha condición e repite unha instrución ata que esa condición deixe de ser certa.
- **break:** funciona como alteración ao funcionamento normal dunha estrutura de repetición, saltándose a condición de funcionamento e pasando a seguinte instrución.

- **Sintaxe avanzada**
  - ♯ **include:** incorpora bibliotecas de funcións que non están incluídas no código actual. Deste xeito, o usuario pode acceder a funcionalidades que non están descritas no sketch.
  - ♯ **define:** proporciona un nome a unha constante antes de que se programe o código.
- **Operadores aritméticos**
  - $=$ **(Asignación):** garda na variable á esquerda do símbolo o que hai á dereita do mesmo.
  - $+$ **(Suma):** devolve a suma aritmética de dous operandos.
- Operadores de comparación
  - $==$**:** Igual que
  - $!=$**:** Distinto de
  - $>$ / $<$**:** Maior / menor que
- Operadores lóxicos
  - $\&\&$: función AND lóxica, devolve un valor verdadeiro se ambas definición son certas.
  - $||$: función OR lóxica, devolve un valor verdadeiro se unha das definicións é certa.
  - !: función NOT lóxica, devolve un valor verdadeiro se a definición é falsa.
- Operadores bit a bit
  - $>>$ / $<<$: despraza a esquerda/ dereita o número de bits que se lle indique á dereita deste operando. Un desprazamento consiste en introducir ceros.

- **Variables**
  - **Tipos de datos**
    - **void:** de uso soamente en definición de funcións, indica que a execución da mesma non retorna ningún valor.
    - **boolean:** ocupa un byte e ten dous posibles valores: true ou false.
    - **char:** ocupa un byte e almacena o número correspondente en código ASCII dun valor. Na definición de variables deste tipo utilízanse comas simples (' ').

- **byte:** garda unha variable de 8 bits sen signo. É dicir un valor entre 0 e 255.
- **int:** almacena variables numéricas de 2 bytes. Pode gardar números negativos, polo que o seu rango de utilización é -32768 a 32767. Os números negativos gárdanse en complemento a dous.
- **String (obxecto):** é unha mellora da función string qué unicamente é un array de datos. Coa definición deste obxecto poden manipularse cadeas de carácteres dun xeito máis especializado. Esta clase contén funcións como:
  - **String.length():** que devolve o tamaño da cadea.
  - **String.concat:** que permite concatenar outras cadeas ou carácteres a continuación da actual.
  - **String.charAt():** que devolve o carácter contido na posición que se indique entre parénteses.

- **Utilidades**
  - **sizeof():** este operador devolve o número de bytes que ocupa unha variable ou un array.

Funcións

- **Dixitais**
  - **pinMode(pin, modo):** configura un dos pines dixitais como entrada (INPUT) ou saída (OUTPUT), facendo cambiar o comportamento eléctrico dese contacto entre alta impedancia ou baixa.
  - **digitalWrite(pin, valor):** naqueles contactos definidos como saída, coloca un nivel alto (HIGH), que pode ser de 5 voltios ou 3,3 dependendo da placa escollida; ou un nivel baixo (LOW), de 0 voltios.
  - **digitalRead(pin):** comproba o valor dun pin avaliándoo como de nivel alto tras exceder un limiar de tensión.
- **Analóxicas**
  - **analogReference(tipo):** serve para definir a referencia interna do convertedor A/D. Pode ter os seguintes argumentos:
    - **DEFAULT:** a tensión correspondente de funcionamento da placa (3,3 ou 5).
    - **INTERNAL:** referencia interna de 1,1 voltios para as placas con procesadores ATmega168 ou ATmega328.
    - **INTERNAL1V1:** referencia interna de 1,1 voltios para a placa MEGA.
    - **INTERNAL2V56:** referencia interna de 2,56 voltios para a placa MEGA.
    - **EXTERNAL:** o voltaxe aplicado no contacto AREF.
  - **analogRead(pin):** converte nun valor entre 0 e 1024 a tensión do contacto indicado. O convertedor integrado ten unha resolución de 4,9 mV por unidade. A frecuencia de mostraxe máxima é de 10000 veces por segundo.

- **analogWrite(pin):** utilizado na función PWM. Crea unha onda cadrada en función do valor de 8 bits que se indica como ciclo de traballo.

- **Temporais**
  - **millis():** devolve o valor en milisegundos que leva funcionando o procesador.
  - **delay(tempo):** pausa o programa durante os milisegundos que se lle indique.
- **Comunicación**
  - **Serial:** esta clase de funcións permite a comunicación entre a placa Arduino e un PC ou outros dispositivos. Todas as placas Arduino teñen unha conexión deste tipo que está incorporada nos contactos dixitais 0 e 1, que a súa vez están imbricados na comunicación vía USB. Pódese acceder a el tanto para ver datos como para envialos a través do monitor serie que inclúe o entorno de desenrolo. Esta clase conten as seguintes funcionalidades que convén coñecer:
    - **Serial.begin(velocidade de transmisión, configuración):** establece os bits por segundo que se van transmitir e a configuración dos mesmos. Este segundo parámetro é opcional e indica paridade dos datos e os bits de stop.
    - **Serial.print/.println(valor):** imprime no monitor serie un texto en formato ASCII. Ao engadírselle ln pasa a seguinte liña en cada impresión.
    - **Serial.write(valor):** escribe un byte no monitor serie.

Estas son todas as funcións propias de Arduino que se utilizarán no deseño, pero como a placa ten funcionalidades externas utilizaranse outras funcións e librerías que se explicarán en vindeiros apartados.

7.2. Módulo de comunicación por radio

A comunicación por radio pode facerse por tres vías:

- Wifi
- GSM
- ZigBee

Comparando as tres, chégase á conclusión que a máis axeitada para o deseño é a opción do ZigBee, xa que é de fácil implementación (comunícase a través dun UART), os seus consumos son ínfimos en comparación coas demais tecnoloxías e o alcance que teñen, dependendo da versión, pode chegar a 1600 metros. Pola súa parte a tecnoloxía wifi, aínda que está moi estendida e a súa utilización é sinxela, presenta un alcance demasiado curto (en torno a 20 metros) en exteriores, ademais que debido á inxente cantidade de redes wifi que se utilizan na actualidade, este espazo radioeléctrico está saturado. En canto á tecnoloxía GSM presenta o hándicap da utilización dunha tarxeta SIM nas comunicacións. Isto implicaría a contratación dun servizo de mensaxería cunha operadora telefónica, o que suporía unha maior inversión.

Os módulos elixidos para esta función son os modelos XBee, baseados neste estándar ZigBee e perfectamente compatibles con outros dispositivos que utilicen este estándar.

Os módulos XBee proporcionan unha solución de conectividade inalámbrica de baixo custo para dispositivos electrónicos. Utilizan o protocolo de rede IEEE 802.15.4 e foron deseñados para aplicacións que requiren alto tráfico de datos, baixa latencia e unha sincronización predicible.

Existen dúas variables dentro de cada serie de XBee: os normais e os PRO. A diferenza entre eles é o alcance, que nos PRO é maior; iso si, a cambio dun consumo maior.

	XBee	**XBee PRO**
Alcance en interior	$>$ 30 m	$>$ 90 m
Alcance en exterior	$>$ 90 m	$>$ 1600 m
Consumo en transmisión	45 mA	250 mA
Consumo en recepción/ espera	50 mA	55 mA

Taboa 7.2.0.1 – Comparativa de modelos de XBee

Aínda que o alcance é un parámetro intrínseco ao módulo, un cambio na antena pode facer que o alcance sexa maior. Existen catro tipos de antenas:

(a) Chip Antenna

(b) Wire Antenna

(c) u.FL

(d) RPSMA

Figura 7.2.0.1 – Tipos de antenas XBee

Tamén comentar que existen dúas series distintas dentro dos módulos XBee: as Series 1 e as Series 2. As escollidas neste caso son as Series 1, que son as máis sinxelas de configurar e cobren perfectamente as necesidades de deseño.

Figura 7.2.0.2 – Pinout XBee

Funcións dos contactos:

Pin	Nome	Descrición
1	VCC	Alimentación de 3,3 V
2	DOUT	Saída de datos do UART
3	DIN /CONFIG	Entrada de datos do UART
4	-	Sen configurar
5	RESET	Reset do módulo XBee
6	PWM0 / RSSI	Saída PWM0 / Indicador de intensidade de sinal na recepción
7	PWM1	Saída PWM1
8	-	Sen conectar
9	DTR / SLEEP_RQ / DI8	Control de modo Sleep/ Entrada dixital 8
10	GND	Terra
11	AD4 / DIO4	Entrada analóxica 4/ Entrada-saída dixital 4
12	CTS / DIO7	Control de borrado do buffer/ Entrada-saída dixital 7
13	ON / SLEEP	Indicador de estado
14	VREF	Tensión de referencia para as entradas analóxicas
15	Associate / AD5 / DIO5	Asociación de módulos/ Entrada analóxica 5/ Entrada- saída dixital 5
16	RTS / AD6 / DIO6	Petición de envío/ Entrada analóxica 6/ Entrada- saída dixital 6
17	AD3 / DIO3	Entrada analóxica 3/ Entrada- saída dixital 3
18	AD2 / DIO2	Entrada analóxica 2/ Entrada- saída dixital 2
19	AD1 / DIO1	Entrada analóxica 1/ Entrada- saída dixital 1
20	AD0 / DIO0	Entrada analóxica 0/ Entrada- saída dixital 0

Taboa 7.2.0.2 – Pinout dos módulos XBee

As conexións mínimas necesarias para o funcionamento de Arduino con estes módulos son VCC, GND, DOUT e DIN.

7.2.1. Funcionamento

- **Comunicación serie**

  Estes módulos conéctanse entre si a través de niveis TTL por un porto serie asíncrono. A través del, o módulo pode comunicarse con calquera UART compatible en lóxica e tensión.

  Os datos entran a través do pin DIN como un sinal serie asíncrono, é dicir, que non vai acompañada dun sinal de reloxo. Este sinal ten que manterse en nivel alto cando non

se están transmitindo datos. Cada paquete de datos esta formado por 10 bits: o bit de comezo, que debe ser de nivel baixo; 8 bits de datos, co bit menos significativo primeiro; e o bit de parada, que ten que ser de nivel alto.

Por exemplo para enviar o número decimal “31” (0001 1111) faríase do seguinte xeito:

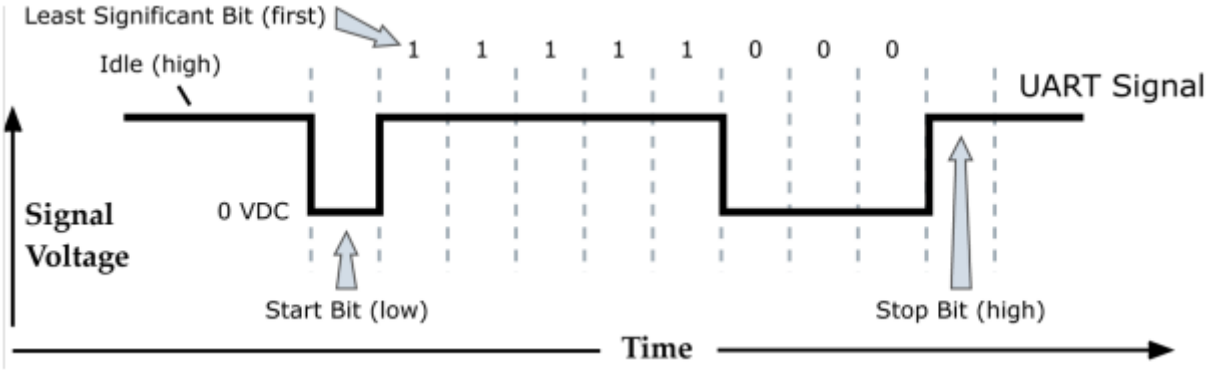


Figura 7.2.1.1 – Funcionamento UART

Como é evidente tanto a UART de orixe como a de destino teñen que estar configuradas do mesmo xeito: taxa de envío, paridade, bits de comezo, bits de parada e bits de datos iguais.

- **Modos de transmisión e recepción**

  - **Modo transparente**

    O modo transparente é o modo máis sinxelo de funcionamento, e non require de ningunha configuración. Na transmisión os datos recibidos por DIN póñense á cola para ser enviados. Na recepción cando se reciben datos mándanse cara o microcontrolador por DOUT.

  - **Modo API**

    Neste modo a información agrúpase en marcos, estendendo de gran forma o nivel de interacción entre dispositivos. Estes marcos conteñen comandos, os datos de envío en si e algunha información sobre o estado da rede. Dende este modo poden ser modificados parámetros de configuración dos dispositivos sen necesidade de entrar no modo de comandos. O seu uso é moi útil para:

    - Transmitir datos a múltiples destinos sen entrar no modo de comandos.
    - Recibir notificacións de que os mensaxes foron recibidos correctamente.
    - Identificar a dirección de orixe dos paquetes de datos recibidos.

  - **Circulación de datos**

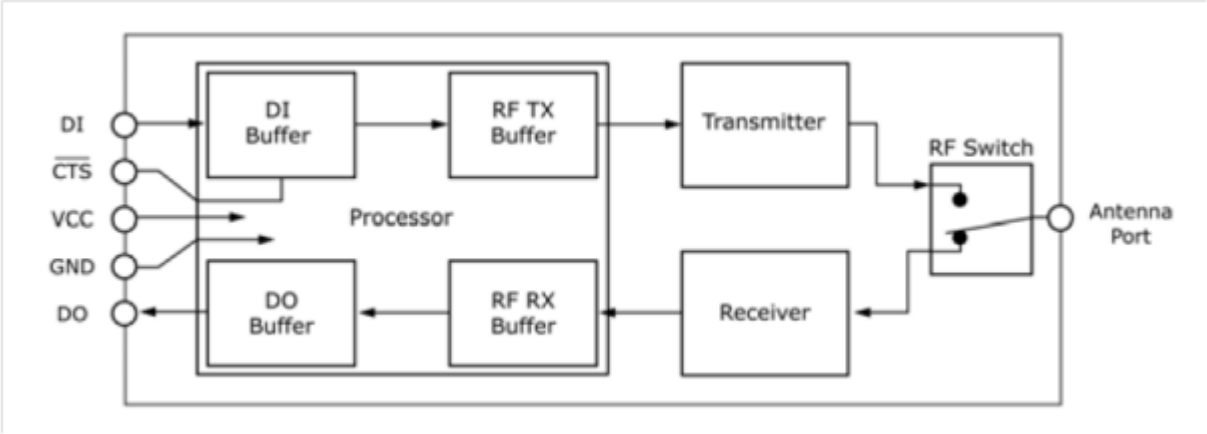


Figura 7.2.1.2 – Circulación de datos

DI Buffer: entrada de datos ao módulo. Cando chegan datos serie ao contacto DI do módulo vanse gardando no DI Buffer ata que podan ser procesados e enviados. Cando este buffer está a 17 bytes de estar completo, o sinal CTS impide que se almacenen máis datos. Esta sinal volve ao seu estado normal cando se recuperan 34 bytes de memoria dispoñibles.

DO Buffer: saída de datos do módulo. No caso da recepción de datos, cada byte que se recibe almacénase no DO Buffer a espera de que sexan requiridos polo dispositivo serie ao que están conectados para colocarse no contacto DO. Cando o DO Buffer alcanza o límite de almacenamento cada novo dato que chegue perderase.

- **Redes XBee**
  - **Peer-to-peer**
    Os módulos XBee, por defecto, están pensados para crear redes ”de igual a igual”, onde todos os dispositivos podan compartir información de xeito que non exista unha relación servidor - cliente. Estas redes presentan a vantaxe de que se manteñen sincronizadas sen necesidade dunha unidade mestra.
    No caso deste deseño, debido a estrutura interna da rede non é posible realizar este tipo de arquitectura, xa que se precisa dun dispositivo que coordine todo o fluxo de datos entre dispositivos.
  - **Redes de asociación**
    Neste tipo de redes existe unha xerarquía mestre- escravo. É polo que aparecen dous tipos de módulos XBee: coordinadores e dispositivos finais. O coordinador é un dispositivo de función completa, que ten a capacidade de establecer todos os parámetros da rede. Pode ter asociado un modo Sleep, ou de baixo consumo, sempre e cando non precise comunicación inmediata cos dispositivos que ten asociados. No caso dos dispositivos finais, son módulos cuxas capacidades están limitadas. Ao formar parte dunha rede xerárquica non son capaces de comunicarse entre eles e unicamente poden facelo co coordinador que teñen

asociado. Esta clase estará a maioría do tempo no modo de baixo consumo, do que sairá ciclicamente para comprobar se ten algunha petición.

- **Direccionamento**

  Os paquetes de datos recibidos vía radio teñen no encabezamento a dirección de orixe e a de destino. Esta dirección pode ser de 16 ou de 64 bits.

  No caso das de 64 bits está definida na fabricación e ten un número único e exclusivo para cada módulo, dividido en dous números de 16 bits (SH e SL). Este dato aparece gravado na parte de atrás dos módulos.

  Para as direccións curtas, de 16 bits, existe un parámetro propio chamado MY, que permite dar aos módulos de forma manual unha dirección.

  - **Modo Unicast**
    Comunicación cos dispositivos de xeito individual. Cando se emite un paquete de datos, os dispositivos que o reciben responden con un ACK, que é un byte que indica que a recepción dos datos foi correcta. En caso de que tras o envío non se reciba esta información, o módulo de xeito automático reenviará os datos tres veces mais ou ata que se reciba. Para a comunicación bidireccional é preciso que coincidan entre os dispositivos o parámetro MY do emisor co DL do receptor.
  - **Modo Broadcast**
    Esta comunicación funciona de xeito que dende un módulo emisor envíanse datos a todos os receptores que estean na rede. Non se contempla o ACK e non existe o reenvío automático.

- **Modos de operación**

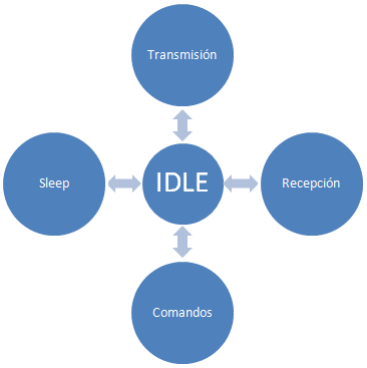


Figura 7.2.1.3 – Modos de operación

  - **Modo IDLE**
    Non se reciben nin se envían datos e estase á espera de que se cumpra algunha das seguintes condicións para cambiar de modo:
    - Recíbense datos serie no buffer DI $\longrightarrow$ Modo transmisión

- ◇ Recíbense datos válidos a través da antena $\longrightarrow$ Modo recepción
- ◇ Comando sleep $\longrightarrow$ Modo sleep
- ◇ Recíbese unha secuencia de comandos $\longrightarrow$ Modo comandos

- **Modos de transmisión e recepción**
  A transmisión de datos pode ser directa ou indirecta. Cando os datos se envían de forma inmediata á dirección de destino é do primeiro tipo. Pola contra cando os datos son retidos durante un período de tempo é do segundo tipo.
- **Modo Sleep**
  Neste modo permítese que o módulo estea "durmindo" mentres nada requira dos seus servizos. Deste xeito éntrase nun modo de moi baixo consumo durante o tempo que non está en uso. A entrada neste modo pode facerse por dúas causas:
  - ◇ Activación do contacto número 9 do módulo XBee.
  - ◇ Supérase o tempo máximo establecido de espera.

  Existen catro variantes no modo Sleep, que se configuran segundo un parámetro interno do módulo denominado SM:

Modo	**Entrada no modo sleep**	**Saída do modo sleep**	**Características**	**Consumo**
Hibernación (SM = 1)	Pin 9 a nivel alto	Pin 9 a nivel baixo	Control por contacto	$< 10\ \mu$A
Durmido (SM = 2)	Pin 9 a nivel alto	Pin 9 a nivel baixo	Control por contacto con espertar rápido	$< 50\ \mu$A
Durmida cíclica (SM = 4)	Transición automática tras pasar un período de tempo definido no parámetro ST	Cando se alcanza o tempo de ciclo de descanso (SP)	O módulo esperta cíclicamente para comprobar se hai datos dispoñibles	$< 50\ \mu$A
Durmida cíclica (SM = 5)	Transición automática tras pasar un período de tempo definido no parámetro ST ou pin 9 a nivel alto	Cando se alcanza o tempo de ciclo de descanso (SP)	O módulo esperta cíclicamente para comprobar se hai datos dispoñibles ou se detecta un flanco de baixada na sinal do pin 9	$< 50\ \mu$A

Taboa 7.2.1.1 – Descrición dos modos Sleep

- **Modo comandos**
  Na configuración de parámetros específicos dos módulos utilizase o modo de comandos, que serve tanto para lelos como para modificalos. Neste modo cada carácter que se reciba interprétase como un comando. Existen dous tipos de modos de comandos: o AT e o API.
  Para entrar no modo de comandos AT hai que enviar a seguinte secuencia a través do porto serie:
  1. Non enviar datos durante un segundo.

2. Enviar a secuencia $+++$ sen esperar.
3. Non enviar datos durante un segundo.

Tras realizar esta secuencia o módulo responderá a través do porto serie cun "OK".

Para enviar comandos AT séguese esta sintaxe:

$$\underset{\text{(Prefixo)}}{\text{AT}} + \text{Comando en ASCII} + \text{Espazo} + \underset{\text{(Opcional)}}{\text{Parámetro en hexadecimal}} + \text{<CR>}$$

Para saír do modo comandos pódese enviar o comando ATCN ou esperar un tempo sen enviar comandos válidos.

7.2.2. Configuración dos módulos

Para definir os parámetros de funcionamento dos emisores de radio frecuencia temos a opción de cambialos a través do envío de comandos polo porto serie ou utilizando o software que proporciona o fabricante a tal efecto.

A forma máis axeitada é a segunda, xa que este software, denominado X-CTU, permite probar as unidades e comprobar o funcionamento da rede.

Existen multitude de parámetros pero os importantes á hora de crear unha rede son os seguintes:

DL *(Destination Address Low)*: define os 32 bits máis baixos dunha dirección de 64 bits. Utilizando o direccionamento en 16 bits, este é o número do módulo de destino da información.

MY *(16-bit Source Address)*: define a dirección de 16 bits persoal que se lle da ao módulo.

CE *(Coordinator Enable)*: 0 se é un dispositivo final, 1 se é o coordinador.

AP *(API Enable)*: Habilita ou deshabilita o modo API. 0 para deshabilitalo, 1 para habilitalo, 2 para habilitalo con caracteres de control de escape.

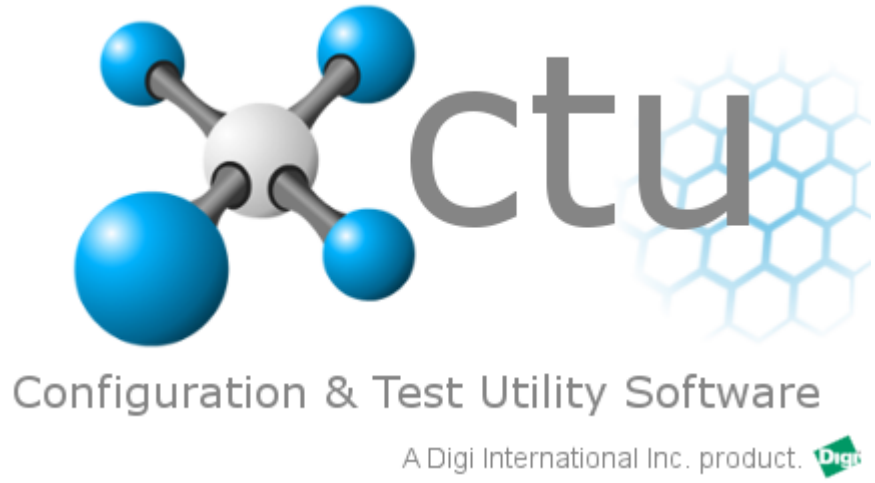
SM *(Sleep Mode)*: Configura o modo de baixo consumo a utilizar.

ST *(Time before sleep)*: Establece o tempo de inactividade (sen fluxo de entrada ou saída de datos RF) que pasará antes de que se entre no modo de descanso. Unicamente é valido nos modos 4 e 5. Este valor debe se igual no coordinador e no dispositivo final.

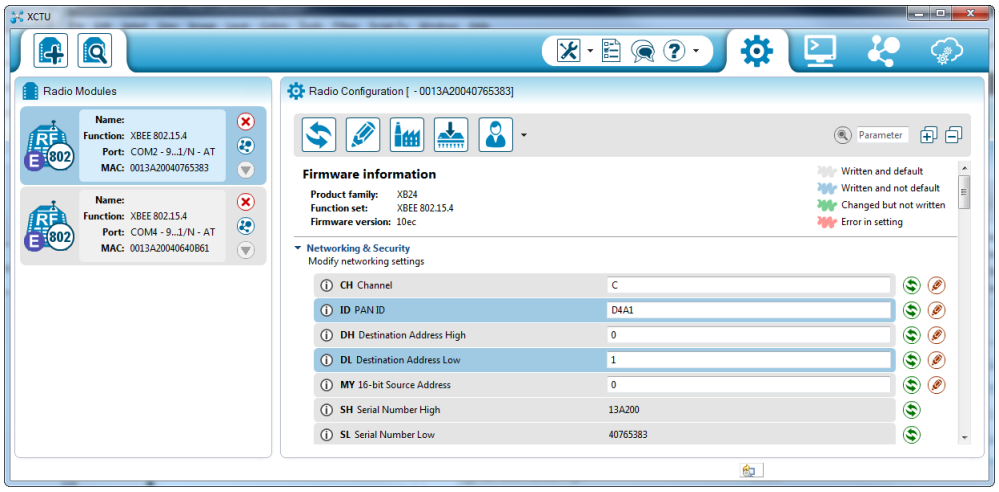
SP *(Cyclic sleep period)*: Establece o tempo máximo por ciclo que o dispositivo estará inactivo. En ambos módulos debe ser igual este valor.

EE *(AES Encryption Enable)*: Permite a encriptación da información.

KY *(AES Encryption Key)*: Contrasinal de 128 bits para encriptar e desencriptar os datos. Este valor non se pode ler.



(a) X-CTU



(b) Pantalla de configuración

Figura 7.2.2.1 – Configuración dos módulos

7.2.3. Funcionamento API

Sen este modo de funcionamento os módulos actuarían como un porto serie inalámbrico, enviando datos de xeito descontrolado. Para evitar este comportamento defínese o modo API, que implementa marcos de información para a súa interpretación correcta.

Especificacións dos marcos API:

Marcador de inicio	Tamaño		Marco de datos	Checksum
Byte 1	Bytes 2 - 3		Bytes 4 - n	Byte n + 1
0x7E	MSB	LSB	Estrutura específica de API	1 Byte

Checksum: este byte permite comprobar a integridade dos datos.

Cálculo: suma de todos os datos do marco de datos quedándose cos últimos 8 bits e restando o resultado a 0xFF (255).

Comprobación: suma dos bytes do marco de datos co checksum. Se o checksum é correcto a suma debe ser 0xFF (255).

Tipos API

Dentro da estrutura específica deste modo existen dous campos de tamaño variable.

- Identificador API
- Datos de identificador

Referíndonos ao deseño descrito nesta documentación, as estruturas empregadas presentan a seguinte forma:

- **Petición de comando AT**
  - **Identificador API:** 0x08
  - **Datos de identificador:**
    - **ID:** identificación do destino da petición para poder enviar o acuse de recibo (ACK).
    - **Comando AT:** Nome do comando con dous carácteres ASCII
    - **Valor de parámetro:** si está presente significa que se lle quere dar un novo valor do comando AT. Se pola contra esta baleiro é que se quere ler o valor actual.
- **Resposta de comando AT**
  - **Identificador API:** 0x88
  - **Datos de identificador:**
    - **ID:** indica que se recibiu a petición con éxito.
    - **Comando AT:** Nome do comando con dous carácteres ASCII.
    - **Estado:**
      - 0 = OK
      - 1 = ERROR
      - 2 = Comando non válido
      - 3 = Parámetro non válido
    - **Valor:** valor hexadecimal do rexistro
- **Petición de transmisión *(Dirección de 16 bits)***
  - **Identificador API**: 0x01
  - **Datos de identificador:**
    - **ID:** identificación do destino da petición para poder enviar o acuse de recibo (ACK).

- **Dirección de destino:** byte máis significativo - byte menos significativo ou envío múltiple (broadcast).
- **Opcións:**
  - 0x01 = deshabilitar ACK
  - 0x04 = Enviar paquete de xeito múltiple
- **Datos RF:** ata 100 bytes por paquete

- **Estado de transmisión**
  - **Identificador API:** 0x89
  - **Datos de identificador:**
    - **ID:** indica que se recibiu a petición con éxito.
    - **Estado:**
      - 0 = Éxito
      - 1 = Non se recibiu ACK. Cando se reintentou todas as veces sen éxito.
      - 2 = Fallo de CCA
      - 3 = Purgado. Cando o coordinador decide borrar os datos que ten á cola debido a que o dispositivo final non saíu do modo Sleep.
- **Recepción de paquete *(Dirección de 16 bits)***
  - **Identificador API:** 0x81
  - **Datos de identificador:**
    - **Dirección de orixe:** MSB - LSB
    - **RSSI:** indicador de potencia de sinal. Valor hexadecimal equivalente a -dBm
    - **Opcións:**
      - bit 0 - Reservado
      - bit 1 = Dirección múltiple
      - bit 2 = Envío múltiple dentro da rede persoal
      - bit 3 a 7 - Reservados
    - **Datos RF:** ata 100 bytes por paquete

7.2.4. Programación propia do modo API

Dada a estrutura especial deste modo de programación requírese dunha biblioteca de funcións que defina e ordene os datos do xeito máis axeitado. Segundo o seu uso as sentencias propias de XBee que se empregan nun sketch de Arduino clasifícanse en:

Sentencias comúns: Presentes en calquera operación con XBee.

1. Inclusión da biblioteca de funcións do fabricante:
2. Creación do obxecto XBee:

3. Inicialización do porto serie e asignación ao módulo XBee.

Código 7.1: Sentencias comúns

```
#include <XBee.h>. //Incorpora as constantes e definicións de funcións deste modo.
XBee <nome> = XBee();
<Nome do porto serie>.begin(velocidade). //Inicializa o porto serie de saída dos datos
<nome do obxecto xbee>.setSerial(Porto serie elixido). //Asigna ese porto serie aos terminais de comunicación do módulo de radio.
```

Lectura de comandos

1. Creación da estrutura de obtención da resposta
2. Permite facer unha solicitude de parámetros.
3. Permite recibir parámetros.
4. Función de lectura de parámetros

Código 7.2: Lectura de comandos

```
XBeeResponse <nome> = XBeeResponse();
ATCommandRequest <nome> = ATCommandRequest() //Array de 8 bits sen signo cos dous caráctreres do nome do parámetro a ler
ATCommandResponse <nome> = ATCommandResponse();
void sendAtCommand() {
    xbee.send(atRequest);
      if (xbee.readPacket(100)) {
        if (xbee.getResponse().getApiId() == AT_COMMAND_RESPONSE) {
           xbee.getResponse().getAtCommandResponse(atResponse);
              if (atResponse.isOk()) {
                  if (atResponse.getValueLength() > 0) {
                     dato_h = (atResponse.getValue()[0]);
                     dato_l = (atResponse.getValue()[1]);
                  }
            }
        }
    }
}
```

En primeiro lugar mediante o comando “xbee.send” envíase a través do contacto TX1 do porto serie de Arduino, cara o contacto DI do XBee o marco API co identificador 0x08 e o valor do parámetro a ler. A continuación lese a resposta e compróbase que o identificador sexa o

dunha resposta a unha petición de comandos AT (0x89) e lese o campo de valores que garda os datos nas variables correspondentes.

Envío de datos RF e comprobación de estado da transmisión.

1. Creación das estruturas de envío dos datos Crea o marco para unha petición de transmisión de datos. Os parámetros desta petición son a dirección de destino, un array de 8 bits sen signo para conter os datos e o tamaño do mesmo.
2. Creación das estruturas de comprobación de estado da transmisión. Marco para recibir o acuse de recibo da información.
3. Función de envío de datos RF

Código 7.3: Envío de datos RF

```
Tx16Request <nome> = Tx16Request (dirección, payload, sizeof(payload));
TxStatusResponse txStatus = TxStatusResponse();
void envio() {
  xbee.send(tx);
  if (xbee.readPacket(5000)) {
    if (xbee.getResponse().getApiId() == TX_STATUS_RESPONSE) {
      xbee.getResponse().getZBTxStatusResponse (txStatus);
      if (txStatus.getStatus() == SUCCESS) {
        ack = true;
      } else {
        ack = false;
      }
    }
  }
}
```

En primeiro lugar a través da instrución “xbee.send” envíase un marco de petición de transmisión, co identificador API 0x01 e os datos a enviar. A continuación compróbase que o marco recibido contén o identificador 0x89 e campo de estado o valor 0, que significa que a recepción se realizou correctamente.

É importante que sempre que se configure no envío de datos a opción de comprobación de ACK se faga a continuación, xa que se non se fai e logo se quere ver a resposta á petición será imposible.

Recepción de datos RF

1. Creación das estruturas para a recepción de datos Crease o marco de recepción de respostas API.
2. Función de lectura de datos RF

Código 7.4: Recepción de datos RF

```
Rx16Response rx16 = Rx16Response();
void lectura() {
  xbee.readPacket();
  if (xbee.getResponse().isAvailable()) {
    if (xbee.getResponse().getApiId() == RX_16_RESPONSE) {
      xbee.getResponse().getRx16Response(rx16);
      data = rx16.getData(0);
    }
  }
}
```

Procédese a almacenar o que hai no buffer de entrada do módulo. Logo compróbase que o identificador é o 0x81 e a continuación extráense os datos necesarios.

7.3. Módulo de comunicación Ethernet

Para realizar a comunicación entre o dispositivo Arduino e unha rede ethernet precísase dunha tarxeta de conexión baseada no chip Wiznet W5100 con un buffer interno de 16k e con un conector RJ-45. Este dispositivo permite crear unha estrutura cliente - servidor que será a que controle o fluxo de datos entre os dispositivos de comunicación.

Para isto asígnase unha dirección IP á placa de Arduino, coa que estará conectado á rede. Así cando dende un navegador web configurado na mesma rede que o Arduino se acceda a esa IP, enviarase unha petición HTTP que será atendida polo servidor, neste caso configurado mediante a tarxeta mencionada e respondida co código HTML correspondente.

Unha petición HTTP ten a seguinte forma:

GET / HTTP/1.1
Host: (dirección IP do servidor)
Accept: text/html
User-Agent: (Navegador e sistema operativo)
[Liña en blanco]

E a resposta do servidor sería deste tipo:

HTTP/1.1 200 OK
Date: (Data e hora da conexión)
Content-Type: text/html
[Contido da páxina HTML]

O dispositivo de conexión entre a rede e Arduino será a Ethernet Shield. Esta extensión permite conectar a placa de programación con internet de xeito fácil e rápido a través dun cable RJ45 e unha secuencia de instrucións simple.

O chip Wiznet W5100 do que dispón, proporciona unha rede IP e é capaz de traballar tanto no modo TCP como no UDP. Pode traballar con 4 conexións simultáneas. Tamén inclúe un lector de tarxetas SD, que pode servir como soporte para gardar rutinas de funcionamento. A conexión á placa Arduino faise a través do estándar SPI que pode encontrarse en varios pines nas placas.

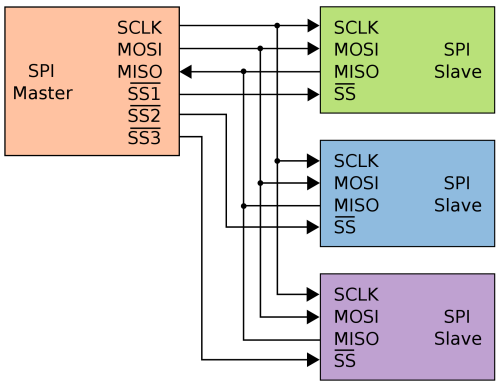
O SPI (Interface de periféricos serie) é un estándar de comunicacións usado na transferencia de información entre circuítos integrados en equipos electrónicos. É un protocolo síncrono e baséase na utilización de 4 sinais:

SCLK *(Serial Clock)*: é o pulso que marca a sincronización. En cada pulso lese ou envíase un bit.

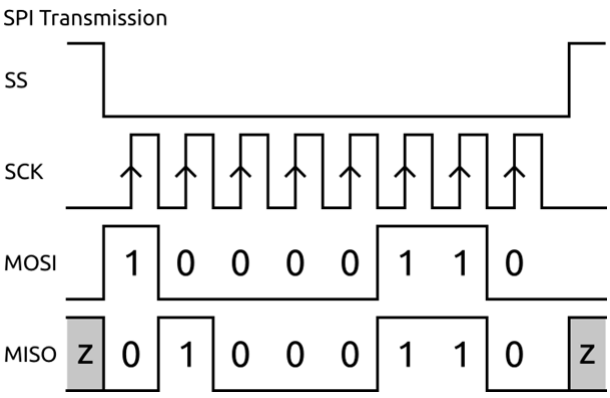
MOSI *(Master Output Slave Input)*: Indica que a dirección de circulación dos datos é do mestre ao escravo.

MISO *(Master Input Slave Output)*: Indica que a dirección de circulación dos datos é do escravo ao mestre.

SS/Select: Selecciona o escravo correspondente e actívao.



(a) Esquema sistema SPI



(b) Transmisión de datos SPI

Figura 7.3.0.1 – SPI

Como axuda para a comprobación do estado de funcionamento da módulo existen 7 LEDs integrados que proporcionan os seguintes datos:

PWR: placa e extensión alimentados correctamente.

LINK: presenza dunha rede pola que comunicarse. Pestanexa cando se emiten ou se reciben datos.

FULLD: conexión de rede full dúplex.

100M: conexión de 100 Mb/s en lugar da de 10 Mb/s.

RX: pestanexa cando a extensión recibe datos.

TX: pestanexa cando a extensión emite datos.

COLL: pestanexa cando se detecta unha colisión na rede (emisión e recepción a un tempo).

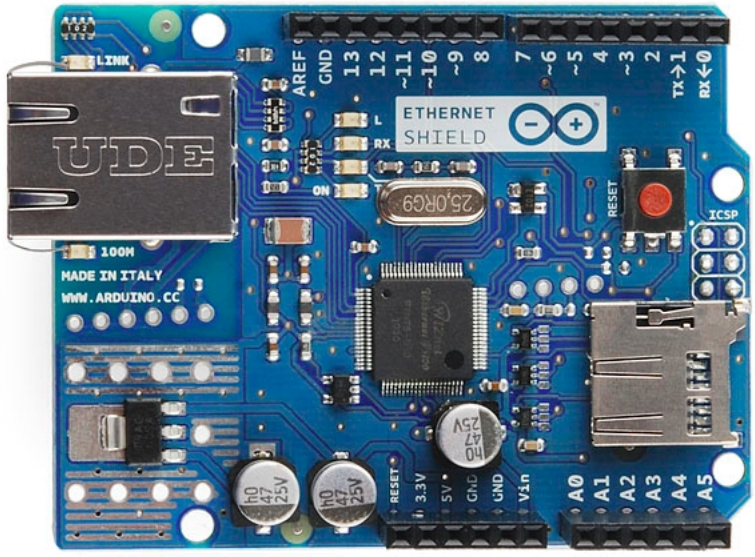


Figura 7.3.0.2 – Ethernet Shield

Por si soa esta extensión é incapaz de proporcionar unha conexión de rede ao deseño. Evidentemente hai que recorrer a un router que nos proporcione a conexión de rede requirida. Para iso hai que realizar unha serie de configuracións que permitan asignar unha dirección IP ao noso dispositivo.

A configuración de cada router é distinta, pero os parámetros a modificar non, polo que a grandes trazos sería deste xeito:

1. Coñecer a porta de enlace do router. Pode ser proporcionada na documentación do mesmo ou pode descubrirse a través do comando “ipconfig” na consola de comandos de Windows.

2. Acceder ao router a través da porta de enlace.

3. Configurar a LAN (Area de conexión local). Por defecto os router asignan direccións IP locais de xeito temporal a cada dispositivo mediante o protocolo DHCP, cada certo tempo cambian e hai que reconectar os dispositivos. No caso deste deseño interesa manter unha mesma IP fixa para o dispositivo. Isto faise a través da MAC do módulo, que é un parámetro que podemos cambiar nos mesmos. Debemos indicarlle ao router que asigne unha dirección IP a esa MAC e que a manteña de forma permanente.

A forma de visualización dos datos, farase nun servidor web (páxina web), que para poder ser creada precisa dun linguaxe de programación propio denominado HTML, que se basea en sentenzas curtas que crean obxectos.

Neste deseño unicamente se contempla o uso dunha caixa de inserción de texto e un botón de envío, pero as posibilidades que ofrece esta linguaxe son moito maiores.

7.3.1. Funcións propias de Ethernet

Como todas as extensións de Arduino, esta tamén ten a súa propia biblioteca de funcións para manexar as variables necesarias na comunicación. A continuación unha breve descrición das utilizadas no presente deseño.

- Inclusión da biblioteca de funcións e definición de datos:

  ♯include <Ethernet.h>

  byte MAC [ ] ={x, x, x, x, x, x} Dirección MAC do módulo

  IPAddress IP, GATEWAY,SUBNET: Dirección IP, porta de enlace e submáscara de rede.

- Inicialización da conexión:

  Ethernet.begin(MAC, IP)

  Arranca a conexión da rede co módulo. Os parámetros que utiliza son os definidos no router.

- Creación de servidor e cliente.

  Na arquitectura da conexión Ethernet aparecen dúas figuras: o cliente e o servidor. O primeiro é o que realiza as peticións de datos e o servidor o que as procesa e as responde en caso de ser posible.

  - EthernetServer <nome do servidor>(porto de conexión)
    Deste xeito créase un servidor que estará escoitando as posibles peticións no porto de conexión. Este acostuma ser por defecto o 80.
  - <nome do servidor>.begin():
    Indica que comeza a escoita de peticións.

- EthernetClient <nome do cliente>
  Crea o cliente que se conectará a unha IP específica e porto.
- <nome do cliente> = <nome do servidor>.available()
  Conecta o cliente co servidor de xeito que podan comezar a comunicarse.
- <nome do cliente>.connected()
  Comproba se o cliente está conectado. En caso de que xa estea desconectado e existan peticións sen ler considerarase que o cliente segue conectado.
- <nome do cliente>.available()
  Devolve o número de bytes dispoñibles para a lectura.
- <nome do cliente>.read()
  Fai a lectura do seguinte byte dispoñible.
- <nome do cliente>.print()
  Serve para imprimir datos sobre o servidor ao que está conectado en formato ASCII.
- <nome do cliente>.println()
  Ao igual que o anterior pero facendo que se comece nunha nova liña. Mediante esta sentenza enviarase o código HTML para ser visualizado no navegador.
- <nome do cliente>.stop()
  Paraliza a conexión entre o cliente e o servidor.

- Exemplo de código

Código 7.5: Exemplo de ethernet

```
if (client) {
  boolean lineaActualEstaVacia = true;
  while (client.connected()) {
    if (client.available() > 0) {
      char c = client.read();
      if (peticion.length() < 15) {
        peticion.concat(c);
      }
    if (c == '\n' && lineaActualEstaVacia) {
      client.println("HTTP/1.1 200 OK");
      client.println("Content-Type: text/html");
      client.println();
      client.println("<!DOCTYPE HTML>");
      client.println("<html>");
```

Neste extracto de código comézase unha páxina HTML. En primeiro lugar compróbase se o cliente está conectado. En caso de que o estea e teña datos dispoñibles comeza a lerse a súa petición. Unha petición finaliza co caracter '\n' , que é un cambio de liña, e unha liña en branco. Cando se detecta esta condición envíase a cabeceira HTML.

8 Resultados finais

En xeral o funcionamento é o seguinte: dende a páxina web mándase unha petición de datos, que cruza o router wifi cara a Ethernet Shield. Nela obtéñense os datos que se lle proporcionaran ao Arduino MEGA, o cal será encargado de procesalos e sacalos en forma de radio frecuencia a través do módulo XBee coordinador. Esta mensaxe envíase ao dispositivo final XBee que se precise, o cal devolverá a información cara o coordinador recorrendo o sentido inverso cara a páxina web.

Na configuración a través do software X-CTU dos módulos XBee utilizaranse os seguintes parámetros:

- **Unidade mestra**
  - **MY:** 1
  - **CE:** 1
  - **AP:** 2
- **Unidades escravas**
  - **DL:** 1
  - **MY:** 2 a 255
  - **CE:** 0
  - **AP:** 2

Os módulos XBee estarán configurados no modo Sleep número 4, que consiste en que entrarán neste modo de baixo consumo ciclicamente. Os tempos que se definen para esta operación terán que ser calculados empiricamente dependendo do uso que vaian ter e a velocidade de resposta que esperemos dos mesmos. Se deles se espera unha resposta inmediata, o ciclo de descanso ten que ser moi curto. Un valor entre os 500 milisegundos e un segundo sería máis que suficiente. Deste xeito cada vez que se pase un tempo definido sen ningún tipo de interacción entre módulos o XBee entrará no modo de descanso, do cal sairá ciclicamente, segundo outro tempo definido; a comprobar se ten algunha petición en espera, a cal procesará.

Dado que estes módulos soportan un cifrado da información aproveitarase esta condición para facer máis segura a comunicación. A través da configuración dos módulos establecerase

un número que actuará como contrasinal na habilitación do modo de cifrado. É importante que este contrasinal se asocie a todos os elementos da rede implicados na comunicación.

Un dos requisitos de deseño era que existise a posibilidade de comunicarse cos módulos máis afastados do mestre da rede que son incapaces de comunicarse de xeito directo con el. Para conseguilo intercalaranse cada certo metros (dependendo do modelo escollido) nodos coordinadores. Deste xeito cando o mestre envíe un paquete de datos cara un elemento remoto e non reciba del o acuse de recibo (ACK), enviará unha solicitude aos demais coordinadores da rede para que intenten establecer a comunicación con el. No caso de que estes coordinadores intermedios tampouco sexan capaces de comunicarse co módulo remoto xerarase unha alarma para indicar que existe un problema con ese módulo en concreto.

Unha das causas que pode facer que o módulo non responda é que se esgote a batería. Como outro dos requisitos mínimos do deseño contémplase o seguimento da fonte de enerxía dos módulos remotos. Esta funcionalidade farase a través dunha das entradas analóxicas libres da plataforma. Dado que se alimentarán cunha tensión por riba dos 5 voltios haberá que incluír un divisor de tensión cuxa saída equivalente oscile entre os 0 e os 5 voltios. Por norma xeral a fonte de enerxía elixida para alimentar os módulos serán baterías de 9 voltios, cun divisor de tensión de 1/2 será suficiente. A interpretación desta lectura farase mediante software e amosará un porcentaxe aproximado de carga das baterías e xerará un aviso cando se atope por debaixo dos 7 voltios (78 %).

Os módulos XBee funcionan con unha tensión de 3,3 voltios de alimentación e teñen uns consumos que poden chegar aos 250 mA. Aínda que as placas de Arduino proporcionan unha saída estable de 3,3 voltios é preferible alimentalos dende unha fonte externa ou dende un regulador sobre a fonte de 5 voltios regulada da placa, xa que a corrente de saída da fonte de 3,3 voltios pode resultar insuficiente e facer que non cheguen a establecerse as comunicacións. Por outro lado debe poder independizarse a conexión dos contactos de comunicación da plataforma Arduino (RX0 e TX1) das do módulo XBee (DIN e DOUT), xa que cos dous módulos conectados e alimentados non é posible a carga dun sketch na memoria da placa de desenrolo, xa que os contactos do porto serie pensados para a conexión USB nese momento están ocupados polo porto serie do módulo de radio, polo tanto ou ben se desconectan ou ben se conectan á inversa (RX0 con DIN e TX1 con DOUT).

Existen produtos no mercado para facilitar estas tarefas. O seu nome comercial é XBee shield e están dispoñibles en varios fabricantes. Incorporan un regulador de tensión de 3,3 voltios que aporta intensidade suficiente para as operacións do módulo de radio e un conmutador dos terminais de comunicación para permitir que o porto serie se ocupe polo XBee ou polo USB.

Todo o que involucra á plataforma Arduino como é a propia placa, as súas extensións, sensores, actuadores e demais están baseados na filosofía do hardware libre. O que significa que os fabricantes aportan toda a documentación necesaria (follas de características, layouts

das placas, modelos de fabricación) para a creación duns modelos propios e funcionais deste hardware.

Aproveitando esta condición e coa finalidade de crear módulos completamente independentes e funcionais, deséñase unha placa de circuíto impreso que aporte unha solución útil para os módulos escravos. Nela inclúese un soporte para a placa Arduino NANO, un zócalo para o módulo XBee, o regulador de tensión de 3,3 voltios xunto cos condensadores de filtro, un diodo LED para a visualización do estado de asociación dos módulos de radio, un conmutador entre o modo de programación USB e a operación XBee e unha serie de conectores para conectar sensores analóxicos, dixitais e elementos de actuación. O esquemático desta placa aparece nos planos.

Cabe destacar que os códigos de programación, tanto para a placa que actúa como escrava como a que actúa como mestra, unicamente son a orientación e a base do código final. Neles unicamente se contempla a funcionalidade para facer a lectura de unha entrada analóxica e ser visualizada na páxina HTML.

A finalidade destes códigos e amosar as funcións básicas de control da rede de xeito que as variacións que se precisen facer neles tan so consistan en engadir instrucións simples e unicamente referidas a placa Arduino, sen modificar o código de XBee e da parte de ethernet. Basicamente as futuras modificacións consistirán en engadir novos códigos de instrución para facer máis dunha lectura e actuar sobre algún elemento externo.

TÍTULO: **DESEÑO DUNHA REDE SEN FÍOS DE TELEMEDIDA**

ANEXOS

PETICIONARIO: **ESCOLA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

DATA: FEBREIRO DE 2016

AUTOR: O ALUMNO

Fdo.: VÍCTOR MANUEL ANIDOS BLANCO

Índice do documento ANEXOS

9 Documentación de partida

9.1. Asignación do TFG

ESCUELA UNIVERSITARIA POLITÉCNICA

ASIGNACIÓN DE TRABAJO FIN DE GRADO

En virtud de la solicitud efectuada por:
En virtud da solicitude efectuada por:

APELLIDOS, NOMBRE: *Anidos Blanco,Víctor Manuel*
APELIDOS E NOME:

DNI: [redacted] **Fecha de Solicitud:** FEB2016
DNI: *Fecha de Solicitude:*

Alumno de esta escuela en la titulación de Grado en Ingeniería en Electrónica Industrial y Automática, se le comunica que la Comisión de Proyectos ha decidido asignarle el siguiente Trabajo Fin de Grado:

O alumno de esta escola na titulación de Grado en Enxeñería en Electrónica Industrial e Automática, comunícaselle que a Comisión de Proxectos ha decidido asignarlle o seguinte Traballo Fin de Grado:

Título T.F.G:Diseño de una red inalámbrica de telemedida

Número TFG: *770G01A98*

TUTOR:(Titor) *Rivas Rodriguez,Juan Manuel*

COTUTOR/CODIRECTOR:

La descripción y objetivos del Trabajo son los que figuran en el reverso de este documento*:*

A descrición e obxectivos do proxecto son os que figuran no reverso deste documento.

Ferrol a Lunes, 1 de Febrero del 2016

Retirei o meu Traballo Fin de Grado o dia _____ de _____ do ano _____

Fdo: Anidos Blanco,Víctor Manuel

DESCRIPCIÓN Y OBJETIVO:Diseño de un protocolo de comunicaciones para realizar medidas a distancia por radio.

Debe estar basado en la plataforma Arduino.

La comunicación se podrá realizar por ZigBee, WiFi o GSM.

Deberá haber una unidad maestra que deberá:
- Establecer la topología de la red.
- Si es necesario un nodo debe de hacer las funciones de repetidor para hacer llegar la informacion al maestro.
- Deberá soportar por lo menos 255 nodos esclavos.
- Cada nodo esclavo deberá poder realizar medidas analógicas (4 canales) como digitales.
- El sistema debe de disponer de una monitorización de su fuente de energía y poder generar alarmas en caso de baja disponibilidad.

9.2. Táboa de comandos XBee

10. XBee Command Reference Tables

Addressing

Addressing Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
DH	**Destination Address High**.Set/Get the upper 32 bits of the 64-bit destination address. When combined with DL, it defines the 64-bit destination address for data transmission. Special definitions for DH and DL include 0x000000000000FFFF (broadcast) and 0x0000000000000000 (coordinator).	CRE	0 - 0xFFFFFFFF	0
DL	**Destination Address Low**. Set/Get the lower 32 bits of the 64-bit destination address. When combined with DH, it defines the 64-bit destination address for data transmissions. Special definitions for DH and DL include 0x000000000000FFFF (broadcast) and 0x0000000000000000 (coordinator).	CRE	0 - 0xFFFFFFFF	0xFFFF(Coordinator) 0 (Router/End Device)
MY	**16-bit Network Address**. Read the 16-bit network address of the module. A value of 0xFFFE means the module has not joined a ZigBee network	CRE	0 - 0xFFFE [read-only]	0xFFFE
MP	1**6-bit Parent Network Address**. Read the 16-bit network address of the module's parent. A value of 0xFFFE means the module does not have a parent.	E	0 - 0xFFFE [read-only]	0xFFFE
NC	**Number of Remaining Children**. Read the number of end device children that can join the device. If NC returns 0, then the device cannot allow any more end device children to join.	CR	0 - MAX_CHILDREN (maximum varies)	read-only
SH	**Serial Number High**. Read the high 32 bits of the module's unique 64-bit address.	CRE	0 - 0xFFFFFFFF [read-only]	factory-set
SL	**Serial Number Low**. Read the low 32 bits of the module's unique 64-bit address.	CRE	0 - 0xFFFFFFFF [read-only]	factory-set
NI	**Node Identifier.** Stores a string identifier. The register only accepts printable ASCII data. In AT Command Mode, a string can not start with a space. A carriage return ends the command. Command will automatically end when maximum bytes for the string have been entered. This string is returned as part of the ND (Node Discover) command. This identifier is also used with the DN (Destination Node) command. In AT command mode, an ASCII comma (0x2C) cannot be used in the NI string	CRE	20-Byte printable ASCII string	ASCII space character (0x20)
SE	**Source Endpoint**. Set/read the ZigBee application layer source endpoint value. This value will be used as the source endpoint for all data transmissions. SE is only supported in AT firmware.The default value 0xE8 (Data endpoint) is the Digi data endpoint	CRE	0 - 0xFF	0xE8
DE	**Destination Endpoint**. Set/read Zigbee application layer destination ID value. This value will be used as the destination endpoint all data transmissions. DE is only supported in AT firmware.The default value (0xE8) is the Digi data endpoint.	CRE	0 - 0xFF	0xE8
CI	**Cluster Identifier**. Set/read Zigbee application layer cluster ID value. This value will be used as the cluster ID for all data transmissions. CI is only supported in AT firmware.The default value0x11 (Transparent data cluster ID).	CRE	0 - 0xFFFF	0x11
NP	**Maximum RF Payload Bytes**. This value returns the maximum number of RF payload bytes that can be sent in a unicast transmission. If APS encryption is used (API transmit option bit enabled), the maximum payload size is reduced by 9 bytes. If source routing is used (AR < 0xFF), the maximum payload size is reduced further. **Note**: NP returns a hexadecimal value. (e.g. if NP returns 0x54, this is equivalent to 84 bytes)	CRE	0 - 0xFFFF	[read-only]
DD	**Device Type Identifier**. Stores a device type value. This value can be used to differentiate different XBee-based devices. Digi reserves the range 0 - 0xFFFFFF. For example, Digi currently uses the following DD values to identify various ZigBee products: 0x30001 - ConnectPort X8 Gateway 0x30002 - ConnectPort X4 Gateway 0x30003 - ConnectPort X2 Gateway 0x30005 - RS-232 Adapter 0x30006 - RS-485 Adapter 0x30007 - XBee Sensor Adapter 0x30008 - Wall Router 0x3000A - Digital I/O Adapter 0x3000B - Analog I/O Adapter 0x3000C - XStick 0x3000F - Smart Plug 0x30011 - XBee Large Display 0x30012 - XBee Small Display	CRE	0 - 0xFFFFFFFF	0x30000

Node types that support the command: C=Coordinator, R=Router, E=End Device

Networking

Networking Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
CH	**Operating Channel**. Read the channel number used for transmitting and receiving between RF modules. Uses 802.15.4 channel numbers. A value of 0 means the device has not joined a PAN and is not operating on any channel.	CRE	**XBee** 0, 0x0B - 0x1A (Channels 11-26) **XBee-PRO (S2)** 0, 0x0B - 0x18 (Channels 11-24) **XBee-PRO (S2B)** 0, 0x0B - 0x19 (Channels 11-25)	[read-only]
ID	**Extended PAN ID**. Set/read the 64-bit extended PAN ID. If set to 0, the coordinator will select a random extended PAN ID, and the router / end device will join any extended PAN ID. Changes to ID should be written to non-volatile memory using the WR command to preserve the ID setting if a power cycle occurs.	CRE	0 - 0xFFFFFFFFFFFFFFFF	0
OP	**Operating Extended PAN ID**. Read the 64-bit extended PAN ID. The OP value reflects the operating extended PAN ID that the module is running on. If ID > 0, OP will equal ID.	CRE	0x01 - 0xFFFFFFFFFFFFFFFF	[read-only]
NH	**Maximum Unicast Hops**. Set / read the maximum hops limit. This limit sets the maximum broadcast hops value (BH) and determines the unicast timeout. The timeout is computed as (50 * NH) + 100 ms. The default unicast timeout of 1.6 seconds (NH=0x1E) is enough time for data and the acknowledgment to traverse about 8 hops.	CRE	0 - 0xFF	0x1E
BH	**Broadcast Hops.** Set/Read the maximum number of hops for each broadcast data transmission. Setting this to 0 will use the maximum number of hops.	CRE	0 - 0x1E	0
OI	**Operating 16-bit PAN ID**. Read the 16-bit PAN ID. The OI value reflects the actual 16-bit PAN ID the module is running on.	CRE	0 - 0xFFFF	[read-only]
NT	**Node Discovery Timeout**. Set/Read the node discovery timeout. When the network discovery (ND) command is issued, the NT value is included in the transmission to provide all remote devices with a response timeout. Remote devices wait a random time, less than NT, before sending their response.	CRE	0x20 - 0xFF [x 100 msec]	0x3C (60d)
NO	**Network Discovery options.** Set/Read the options value for the network discovery command. The options bitfield value can change the behavior of the ND (network discovery) command and/or change what optional values are returned in any received ND responses or API node identification frames. Options include: 0x01 = Append DD value (to ND responses or API node identification frames) 002 = Local device sends ND response frame when ND is issued.	CRE	0 - 0x03 [bitfield]	0
SC	**Scan Channels**. Set/Read the list of channels to scan. ***Coordinator*** - Bit field list of channels to choose from prior to starting network. ***Router/End Device*** - Bit field list of channels that will be scanned to find a Coordinator/ Router to join. Changes to SC should be written using WR command to preserve the SC setting if a power cycle occurs. Bit (Channel): 0 (0x0B) 4 (0x0F) 8 (0x13) 12 (0x17) 1 (0x0C) 5 (0x10) 9 (0x14) 13 (0x18) 2 (0x0D) 6 (0x11) 10 (0x15) 14 (0x19) 3 (0x0E) 7 (0x12) 11 (0x16) 15 (0x1A)	CRE	**XBee** 1 - 0xFFFF [bitfield] **XBee-PRO (S2)** 1 - 0x3FFF [bitfield] (bits 14, 15 not allowed) **XBee-PRO (S2B)** 1-0x7FFF (bit 15 is not allowed)	1FFE
SD	**Scan Duration**. Set/Read the scan duration exponent. Changes to SD should be written using WR command. ***Coordinator*** - Duration of the Active and Energy Scans (on each channel) that are used to determine an acceptable channel and Pan ID for the Coordinator to startup on. ***Router / End Device*** - Duration of Active Scan (on each channel) used to locate an available Coordinator / Router to join during Association. Scan Time is measured as:(# Channels to Scan) * (2 ^ SD) * 15.36ms - The number of channels to scan is determined by the SC parameter. The XBee can scan up to 16 channels (SC = 0xFFFF). Sample Scan Duration times (13 channel scan): If SD = 0, time = 0.200 sec SD = 2, time = 0.799 sec SD = 4, time = 3.190 sec SD = 6, time = 12.780 sec **Note**: SD influences the time the MAC listens for beacons or runs an energy scan on a given channel. The SD time is not a good estimate of the router/end device joining time requirements. ZigBee joining adds additional overhead including beacon processing on each channel, sending a join request, etc. that extend the actual joining time.	CRE	0 - 7 [exponent]	3
ZS	**ZigBee Stack Profile**. Set / read the ZigBee stack profile value. This must be set the same on all devices that should join the same network.	CRE	0 - 2	0

Networking Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
NJ	**Node Join Time**. Set/Read the time that a Coordinator/Router allows nodes to join. This value can be changed at run time without requiring a Coordinator or Router to restart. The time starts once the Coordinator or Router has started. The timer is reset on power-cycle or when NJ changes.	CR	0 - 0xFF [x 1 sec]	0xFF (always allows joining)
JV	**Channel Verification**. Set/Read the channel verification parameter. If JV=1, a router will verify the coordinator is on its operating channel when joining or coming up from a power cycle. If a coordinator is not detected, the router will leave its current channel and attempt to join a new PAN. If JV=0, the router will continue operating on its current channel even if a coordinator is not detected.	R	0 - Channel verification disabled 1 - Channel verification enabled	0
NW	**Network Watchdog Timeout**. Set/read the network watchdog timeout value. If NW is set > 0, the router will monitor communication from the coordinator (or data collector) and leave the network if it cannot communicate with the coordinator for 3 NW periods. The timer is reset each time data is received from or sent to a coordinator, or if a many-to-one broadcast is received.	R	0 - 0x64FF [x 1 minute] (up to over 17 days)	0 (disabled)
JN	**Join Notification**. Set / read the join notification setting. If enabled, the module will transmit a broadcast node identification packet on power up and when joining. This action blinks the Associate LED rapidly on all devices that receive the transmission, and sends an API frame out the UART of API devices. This feature should be disabled for large networks to prevent excessive broadcasts.	RE	0 - 1	0
AR	**Aggregate Routing Notification**. Set/read time between consecutive aggregate route broadcast messages. If used, AR should be set on only one device to enable many-to-one routing to the device. Setting AR to 0 only sends one broadcast	CR	0 - 0xFF	0xFF

Security

Security Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
EE	**Encryption Enable**. Set/Read the encryption enable setting.	CRE	0 - Encryption disabled 1 - Encryption enabled	0
EO	**Encryption Options.** Configure options for encryption. Unused option bits should be set to 0. Options include: 0x01 - Send the security key unsecured over-the-air during joins 0x02 - Use trust center (coordinator only	CRE	0 - 0xFF	
NK	**Network Encryption Key**. Set the 128-bit AES network encryption key. This command is write-only; NK cannot be read. If set to 0 (default), the module will select a random network key.	C	128-bit value	0
KY	**Link Key**. Set the 128-bit AES link key. This command is write only; KY cannot be read. Setting KY to 0 will cause the coordinator to transmit the network key in the clear to joining devices, and will cause joining devices to acquire the network key in the clear when joining.	CRE	128-bit value	0

RF Interfacing

RF Interfacing Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
PL	**Power Level**. Select/Read the power level at which the RF module transmits conducted power. For XBee-PRO (S2B) Power Level 4 is calibrated and the other power levels are approximate.	CRE	**XBee** (boost mode disabled) 0 = -8 dBm 1 = -4 dBm 2 = -2 dBm 3 = 0 dBm 4 = +2 dBm **XBee-PRO (S2)** 4 = 17 dBm **XBee-PRO (S2)** **(International Variant)** 4 = 10dBm **XBee-PRO (S2B)** (Boost mode enabled) 4 = 18dBM 3 = 16dBm 2 = 14dBm 1 = 12dBm 0 = 10dBm **XBee-PRO (S2B)** **(International Variant)** (Boost mode enabled) 4 = 10dBm 3 = 8dBm 2 = 6dBm 1 = 4dBm 0 = 2dBm	4
PM	**Power Mode**. Set/read the power mode of the device. Enabling boost mode will improve the receive sensitivity by 1dB and increase the transmit power by 2dB Note: Enabling boost mode on the XBee-PRO (S2) will not affect the output power. Boost mode imposes a slight increase in current draw. See section 1.2 for details.	CRE	0-1, 0= -Boost mode disabled, 1= Boost mode enabled.	1
DB	**Received Signal Strength**. This command reports the received signal strength of the last received RF data packet. The DB command only indicates the signal strength of the last hop. It does not provide an accurate quality measurement for a multihop link. DB can be set to 0 to clear it. The DB command value is measured in -dBm. For example if DB returns 0x50, then the RSSI of the last packet received was -80dBm. As of 2x6x firmware, the DB command value is also updated when an APS acknowledgment is received.	CRE	0 - 0xFF Observed range for XBee-PRO: 0x1A - 0x58 For XBee: 0x 1A - 0x5C	
PP	**Peak Power.** Read the dBm output when maximum power is selected (PL4).	CRE	0x0-0x12	[read only]

1. Node types that support the command: C = Coordinator, R = Router, E = End Device

Serial Interfacing (I/O)

Serial Interfacing Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
AP	API Enable. Enable API Mode. The AP command is only supported when using API firmware: 21xx (API coordinator), 23xx (API router), 29xx (API end device).	CRE	1 - 2 1 = API-enabled 2 = API-enabled (w/escaped control characters)	1
AO	**API Options**. Configure options for API. Current options select the type of receive API frame to send out the Uart for received RF data packets.	CRE	0 - Default receive API indicators enabled 1 - Explicit Rx data indicator API frame enabled (0x91) 3 - enable ZDO passthrough of ZDO requests to the UART which are not supported by the stack, as well as Simple_Desc_req, Active_EP_req, and Match_Desc_req.	0
BD	**Interface Data Rate**. Set/Read the serial interface data rate for communication between the module serial port and host. Any value above 0x07 will be interpreted as an actual baud rate. When a value above 0x07 is sent, the closest interface data rate represented by the number is stored in the BD register.	CRE	0 - 7 (standard baud rates) 0 = 1200 bps 1 = 2400 2 = 4800 3 = 9600 4 = 19200 5 = 38400 6 = 57600 7 = 115200 0x80 - 0xE1000 (non-standard rates up to 921kbps)	3
NB	**Serial Parity**. Set/Read the serial parity setting on the module.	CRE	0 = No parity 1 = Even parity 2 = Odd parity 3 = Mark parity	0
SB	**Stop Bits**. Set/read the number of stop bits for the UART. (Two stop bits are not supported if mark parity is enabled.)	CRE	0 = 1 stop bit 1 = 2 stop bits	0
RO	**Packetization Timeout**. Set/Read number of character times of inter-character silence required before packetization. Set (RO=0) to transmit characters as they arrive instead of buffering them into one RF packet The RO command is only supported when using AT firmware: 20xx (AT coordinator), 22xx (AT router), 28xx (AT end device).	CRE	0 - 0xFF [x character times]	3
D7	**DIO7 Configuration**. Select/Read options for the DIO7 line of the RF module.	CRE	0 = Disabled 1 = CTS Flow Control 3 = Digital input 4 = Digital output, low 5 = Digital output, high 6 = RS-485 transmit enable (low enable) 7 = RS-485 transmit enable (high enable)	1
D6	**DIO6 Configuration.** Configure options for the DIO6 line of the RF module.	CRE	0 = Disabled 1 = RTS flow control 3 = Digital input 4 = Digital output, low 5 = Digital output, high	0

1. Node types that support the command: C = Coordinator, R = Router, E = End Device

I/O Commands

I/O Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
IR	**IO Sample Rate**. Set/Read the IO sample rate to enable periodic sampling. For periodic sampling to be enabled, IR must be set to a non-zero value, and at least one module pin must have analog or digital IO functionality enabled (see D0-D8, P0-P2 commands). The sample rate is measured in milliseconds.	CRE	0, 0x32:0xFFFF (ms)	0
IC	I**O Digital Change Detection**. Set/Read the digital IO pins to monitor for changes in the IO state. IC works with the individual pin configuration commands (D0-D8, P0-P2). If a pin is enabled as a digital input/output, the IC command can be used to force an immediate IO sample transmission when the DIO state changes. IC is a bitmask that can be used to enable or disable edge detection on individual channels. Unused bits should be set to 0. Bit (IO pin): 0 (DIO0)4 (DIO4)8 (DIO8) 1 (DIO1) 5 (DIO5) 9 (DIO9) 2 (DIO2) 6 (DIO6) 10 (DIO10) 3 (DIO3) 7 (DIO7) 11 (DIO11)	CRE	: 0 - 0xFFFF	0
P0	**PWM0 Configuration**. Select/Read function for PWM0.	CRE	0 = Disabled 1 = RSSI PWM 3 - Digital input, monitored 4 - Digital output, default low 5 - Digital output, default high	1
P1	**DIO11 Configuration**. Configure options for the DIO11 line of the RF module.	CRE	0 - Unmonitored digital input 3- Digital input, monitored 4- Digital output, default low 5- Digital output, default high	0
P2	**DIO12 Configuration**. Configure options for the DIO12 line of the RF module.	CRE	0 - Unmonitored digital input 3- Digital input, monitored 4- Digital output, default low 5- Digital output, default high	0
P3	**DIO13 Configuration**. Set/Read function for DIO13. This command is not yet supported.	CRE	0, 3-5 0 – Disabled 3 – Digital input 4 – Digital output, low 5 – Digital output, high	
D0	**AD0/DIO0 Configuration**. Select/Read function for AD0/DIO0.	CRE	1 - Commissioning button enabled 2 - Analog input, single ended 3 - Digital input 4 - Digital output, low 5 - Digital output, high	1
D1	**AD1/DIO1 Configuration**. Select/Read function for AD1/DIO1.	CRE	0, 2-5 0 – Disabled 2 - Analog input, single ended 3 – Digital input 4 – Digital output, low 5 – Digital output, high	0
D2	AD2/DIO2 Configuration. Select/Read function for AD2/DIO2.	CRE	0, 2-5 0 – Disabled 2 - Analog input, single ended 3 – Digital input 4 – Digital output, low 5 – Digital output, high	0

I/O Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
D3	**AD3/DIO3 Configuration**. Select/Read function for AD3/DIO3.	CRE	0, 2-5 0 – Disabled 2 - Analog input, single ended 3 – Digital input 4 – Digital output, low 5 – Digital output, high	0
D4	**DIO4 Configuration**. Select/Read function for DIO4.	CRE	0, 3-5 0 – Disabled 3 – Digital input 4 – Digital output, low 5 – Digital output, high	0
D5	**DIO5 Configuration**. Configure options for the DIO5 line of the RF module.	CRE	0 = Disabled 1 = Associated indication LED 3 = Digital input 4 = Digital output, default low 5 = Digital output, default high	1
D8	**DIO8 Configuration**. Set/Read function for DIO8. This command is not yet supported.	CRE	0, 3-5 0 – Disabled 3 – Digital input 4 – Digital output, low 5 – Digital output, high	
LT	**Assoc LED Blink Time**. Set/Read the Associate LED blink time. If the Associate LED functionality is enabled (D5 command), this value determines the on and off blink times for the LED when the module has joined a network. If LT=0, the default blink rate will be used (500ms coordinator, 250ms router/end device). For all other LT values, LT is measured in 10ms.	CRE	0, 0x0A - 0xFF (100 - 2550 ms)	0
PR	**Pull-up Resistor**. Set/read the bit field that configures the internal pull-up resistor status for the I/O lines. "1" specifies the pull-up resistor is enabled. "0" specifies no pullup.(30k pull-up resistors) Bits:" 0 - DIO4 (Pin 11) 1 - AD3 / DIO3 (Pin 17) 2 - AD2 / DIO2 (Pin 18) 3 - AD1 / DIO1 (Pin 19) 4 - AD0 / DIO0 (Pin 20) 5 - RTS / DIO6 (Pin 16) 6 - DTR / Sleep Request / DIO8 (Pin 9) 7 - DIN / Config (Pin 3) 8 - Associate / DIO5 (Pin 15) 9 - On/Sleep / DIO9 (Pin 13) 10 - DIO12 (Pin 4) 11 - PWM0 / RSSI / DIO10 (Pin 6) 12 - PWM1 / DIO11 (Pin 7) 13 - CTS / DIO7 (Pin 12)	CRE	0 - 0x3FFF	0 - 0x1FFF
RP	**RSSI PWM Timer.** Time the RSSI signal will be output on the PWM after the last RF data reception or APS acknowledgment.. When RP = 0xFF, output will always be on.	CRE	0 - 0xFF [x 100 ms]	0x28 (40d)
%V	**Supply Voltage.** Reads the voltage on the Vcc pin. Scale by 1200/1024 to convert to mV units. For examplee, a %V reading of 0x900 (2304 decimal) represents 2700mV or 2.7OV.	CRE	-0x-0xFFFF [read only]	-
V+	**Voltage Supply Monitoring.** The voltage supply threshold is set with the V+ command. If the measured supply voltage falls below or equal to this threshold, the supply voltage will be included in the IO sample set. V+ is set to 0 by default (do not include the supply voltage). Scale mV units by 1024/1200 to convert to internal units. For example, for a 2700mV threshold enter 0x900. Given the operating Vcc ranges for different platforms, and scaling by 1024/1200, the useful parameter ranges are: XBee 2100-3600 mV 0,0x0700-0x0c00 PRO 3000-3400 mV, 0,0x0a00-0x0b55 S2B 2700-3600 mV, 0,0x0900-0x0c00	CRE	0-0xFFFF	0
TP	Reads the module temperature in Degrees Celsius. Accuracy +/- 7 degrees. 1° C = 0x0001 and -1° C = 0xFFFF. Command is only available in PRO S2B.	CRE	0x0-0xFFFF	-

Diagnostics

Diagnostics Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
VR	**Firmware Version**. Read firmware version of the module. The firmware version returns 4 hexadecimal values (2 bytes) "ABCD". Digits ABC are the main release number and D is the revision number from the main release. "B" is a variant designator. XBee and XBee-PRO ZB modules return: 0x2xxx versions. XBee and XBee-PRO ZNet modules return: 0x1xxx versions. ZNet firmware is not compatible with ZB firmware.	CRE	0 - 0xFFFF [read-only]	Factory-set
HV	**Hardware Version**. Read the hardware version of the module.version of the module. This command can be used to distinguish among different hardware platforms. The upper byte returns a value that is unique to each module type. The lower byte indicates the hardware revision. XBee ZB and XBee ZNet modules return the following (hexadecimal) values: 0x19xx - XBee module 0x1Axx - XBee-PRO module	CRE	0 - 0xFFFF [read-only]	Factory-set
AI	**Association Indication**. Read information regarding last node join request: 0x00 - Successfully formed or joined a network. (Coordinators form a network, routers and end devices join a network.) 0x21 - Scan found no PANs 0x22 - Scan found no valid PANs based on current SC and ID settings 0x23 - Valid Coordinator or Routers found, but they are not allowing joining (NJ expired) 0x24 - No joinable beacons were found 0x25 - Unexpected state, node should not be attempting to join at this time 0x27 - Node Joining attempt failed (typically due to incompatible security settings) 0x2A - Coordinator Start attempt failed' 0x2B - Checking for an existing coordinator 0x2C - Attempt to leave the network failed 0xAB - Attempted to join a device that did not respond. 0xAC - Secure join error - network security key received unsecured 0xAD - Secure join error - network security key not received 0xAF - Secure join error - joining device does not have the right preconfigured link key 0xFF - Scanning for a ZigBee network (routers and end devices) **Note**: New non-zero AI values may be added in later firmware versions. Applications should read AI until it returns 0x00, indicating a successful startup (coordinator) or join (routers and end devices)	CRE	0 - 0xFF [read-only]	--

1. Node types that support the command:C = Coordinator, R = Router, E = End Device

AT Command Options

AT Command Options Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
CT	**Command Mode Timeout.** Set/Read the period of inactivity (no valid commands received) after which the RF module automatically exits AT Command Mode and returns to Idle Mode.	CRE	2 - 0x028F [x 100 ms]	0x64 (100d)
CN	**Exit Command Mode.** Explicitly exit the module from AT Command Mode.	CRE	--	--
GT	**Guard Times**. Set required period of silence before and after the Command Sequence Characters of the AT Command Mode Sequence (GT + CC + GT). The period of silence is used to prevent inadvertent entrance into AT Command Mode.	CRE	1 - 0x0CE4 [x 1 ms] (max of 3.3 decimal sec)	0x3E8 (1000d)
CC	**Command Sequence Character**. Set/Read the ASCII character value to be used between Guard Times of the AT Command Mode Sequence (GT + CC + GT). The AT Command Mode Sequence enters the RF module into AT Command Mode. The CC command is only supported when using AT firmware: 20xx (AT coordinator), 22xx (AT router), 28xx (AT end device).	CRE	0 - 0xFF	0x2B ('+' ASCII)

1. Node types that support the command: C = Coordinator, R = Router, E = End Device

Sleep Commands

Sleep Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
SM	**Sleep Mode** Sets the sleep mode on the RF module. An XBee loaded with router firmware can be configured as either a router (SM set to 0) or an end device (SM > 0). Changing a device from a router to an end device (or vice versa) forces the device to leave the network and attempt to join as the new device type when changes are applied.	RE	0-Sleep disabled (router) 1-Pin sleep enabled 4-Cyclic sleep enabled 5 - Cyclic sleep, pin wake	0 - Router 4 - End Device
SN	**Number of Sleep Periods.** Sets the number of sleep periods to not assert the On/Sleep pin on wakeup if no RF data is waiting for the end device. This command allows a host application to sleep for an extended time if no RF data is present	CRE	1 - 0xFFFF	1
SP	**Sleep Period.** This value determines how long the end device will sleep at a time, up to 28 seconds. (The sleep time can effectively be extended past 28 seconds using the SN command.) On the parent, this value determines how long the parent will buffer a message for the sleeping end device. It should be set at least equal to the longest SP time of any child end device.	CRE	0x20 - 0xAF0 x 10ms (Quarter second resolution)	0x20
ST	**Time Before Sleep** Sets the time before sleep timer on an end device.The timer is reset each time serial or RF data is received. Once the timer expires, an end device may enter low power operation. Applicable for cyclic sleep end devices only.	E	1 - 0xFFFE (x 1ms)	0x1388 (5 seconds)
SO Command	**Sleep Options**. Configure options for sleep. Unused option bits should be set to 0. Sleep options include: 0x02 - Always wake for ST time 0x04 - Sleep entire SN * SP time Sleep options should not be used for most applications. See chapter 6 for more information.	E	0 - 0xFF	0
WH	**Wake Host**. Set/Read the wake host timer value. If the wake host timer is set to a non-zero value, this timer specifies a time (in millisecond units) that the device should allow after waking from sleep before sending data out the UART or transmitting an IO sample. If serial characters are received, the WH timer is stopped immediately.	E	0 - 0xFFFF (x 1ms)	
SI	**Sleep Immediately**. See Execution Commands table below..			
PO	**Polling Rate**. Sets the polling rate for the end device.	E	0 - 0x1770 (10msec)	0x00 (100 msec)

Execution Commands

Where most AT commands set or query register values, execution commands cause an action to be executed on the module. Execution commands are executed immediately and do not require changes to be applied.

Execution Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
AC	**Apply Changes**. Applies changes to all command registers causing queued command register values to be applied. For example, changing the serial interface rate with the BD command will not change the UART interface rate until changes are applied with the AC command. The CN command and 0x08 API command frame also apply changes.	CRE	-	
WR	**Write**. Write parameter values to non-volatile memory so that parameter modifications persist through subsequent resets. Note: Once WR is issued, no additional characters should be sent to the module until after the "OK\r" response is received. The WR command should be used sparingly. The EM250 supports a limited number of write cycles.“	CRE	--	--
RE	**Restore Defaults**. Restore module parameters to factory defaults.	CRE	--	--
FR	**Software Reset.** Reset module. Responds immediately with an OK status, and then performs a software reset about 2 seconds later.	CRE	--	--
NR	**Network Reset**. Reset network layer parameters on one or more modules within a PAN. Responds immediately with an “OK” then causes a network restart. All network configuration and routing information is consequently lost. *If NR = 0*: Resets network layer parameters on the node issuing the command. *If NR = 1*: Sends broadcast transmission to reset network layer parameters on all nodes in the PAN.	CRE	0 - 1	--
SI	**Sleep Immediately**. Cause a cyclic sleep module to sleep immediately rather than wait for the ST timer to expire.	E	-	-
CB	**Commissioning Pushbutton**. This command can be used to simulate commissioning button presses in software. The parameter value should be set to the number of button presses to be simulated. For example, sending the ATCB1 command will execute the action associated with 1 commissioning button press.	CRE		

Execution Commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
ND	**Node Discover.** Discovers and reports all RF modules found. The following information is reported for each module discovered. MY<CR> SH<CR> SL<CR> NI<CR> (Variable length) PARENT_NETWORK ADDRESS (2 Bytes)<CR> DEVICE_TYPE<CR> (1 Byte: 0=Coord, 1=Router, 2=End Device) STATUS<CR> (1 Byte: Reserved) PROFILE_ID<CR> (2 Bytes) MANUFACTURER_ID<CR> (2 Bytes) <CR> After (NT * 100) milliseconds, the command ends by returning a <CR>. ND also accepts a Node Identifier (NI) as a parameter (optional). In this case, only a module that matches the supplied identifier will respond. If ND is sent through the API, each response is returned as a separate AT_CMD_Response packet. The data consists of the above listed bytes without the carriage return delimiters. The NI string will end in a "0x00" null character. The radius of the ND command is set by the BH command.	CRE	optional 20-Byte NI or MY value	--
DN	**Destination Node.** Resolves an NI (Node Identifier) string to a physical address (case-sensitive). The following events occur after the destination node is discovered: <AT Firmware> 1. DL & DH are set to the extended (64-bit) address of the module with the matching NI (Node Identifier) string. 2. OK (or ERROR)\r is returned. 3. Command Mode is exited to allow immediate communication <API Firmware> 1. The 16-bit network and 64-bit extended addresses are returned in an API Command Response frame. If there is no response from a module within (NT * 100) milliseconds or a parameter is not specified (left blank), the command is terminated and an “ERROR” message is returned. In the case of an ERROR, Command Mode is not exited. The radius of the DN command is set by the BH command.	CRE	up to 20-Byte printable ASCII string	--
IS	**Force Sample** Forces a read of all enabled digital and analog input lines.	CRE	--	--
1S	**XBee Sensor Sample**. Forces a sample to be taken on an XBee Sensor device. This command can only be issued to an XBee sensor device using an API remote command.	RE	-	-

Node types that support the command: C = Coordinator, R = Router, E = End Device

10 Códigos de programación

Código 10.1: Arduino MEGA - Coordinador da rede

```
1  //Arduino MEGA - Coordinador da rede
2
3  #include <Ethernet.h>     //Librería coas funcións de Ethernet
4  #include <SPI.h>     //Librería de comunicacións via SPI
5  #include <XBee.h>     //Libreía de funcións XBee
6
7  byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};     //Dirección MAC da Ethernet
       Shield
8  IPAddress ip(192,168,0,177);     //Dirección IP da shield
9  IPAddress gateway(192, 168, 0, 1);     //Porta de enlace do router
10 IPAddress subnet(255, 255, 255, 0);     //Submáscara da rede
11 EthernetServer server(80);     //Definición do servidor Ethernet
12
13 //Declaración de variables
14 boolean lineaActualEstaVacia = true;
15 boolean ack;
16 String peticion =" ";
17 int dato = 0;
18 int pos = 0;
19 int pos1 = 0;
20 int pos2 = 0;
21 int pos3 = 0;
22 int pos4 = 0;
23
24 XBee xbee = XBee();     //Declaración do obxecto XBee
25
26 //uint8_t -> 8 bits sin signo
27 uint8_t payload[] = {0};
28
29 uint8_t data_l = 0;
30 uint8_t data_h = 0;
31 uint8_t id_l = 0;
32 uint8_t id_h = 0;
33
34 int dir = 0x0000;
```

```
int car_ASCII[] = {0, 0, 0};
int car_dec = 0;

//Variables propias de XBee
Tx16Request tx = Tx16Request (dir, payload, sizeof(payload));
TxStatusResponse txStatus = TxStatusResponse();
Rx16Response rx16 = Rx16Response();

/*Instrucciónsdearranquedosistema:
  1–Inicialización dos portos serie coa velocidade de transmisión
  2–Asignación do porto serie ao módulo de radio
  3–Asignación de direccións de uso do módulo Ethernet
  4–Temporización
*/

void setup() {

  Serial.begin(9600);
  Serial3.begin(9600);
  xbee.setSerial(Serial3);
  Ethernet.begin(mac, ip);
  server.begin();

  delay(1000);

}


//Lectura de paquetes de datos RF
void lectura() {
  delay(100);
  xbee.readPacket();
  if (xbee.getResponse().isAvailable()) {
    if (xbee.getResponse().getApiId() == RX_16_RESPONSE) {
      xbee.getResponse().getRx16Response(rx16);
      id_h = rx16.getData(3);
      id_l = rx16.getData(2);
      data_l = rx16.getData(1);
      data_h = rx16.getData(0);
      }
     }

 }

//Envío de datos RF
void envio() {
  tx = Tx16Request (dir, payload, sizeof(payload));
  xbee.send(tx);
  if (xbee.readPacket(5000)) {
      if (xbee.getResponse().getApiId() == TX_STATUS_RESPONSE) {
```

```
85          xbee.getResponse().getZBTxStatusResponse (txStatus);
86          if (txStatus.getStatus() == SUCCESS) {
87            ack = true;
88            } else {
89            ack = false;
90            }
91          }
92       }
93 }
94
95 /*Bucle principal de funcionamento:
96   1- Compróbase que hai unha petición dun cliente
97   2- En caso de habelo notifícase no monitor serie
98   3- Lese a peticion do cliente
99   4- Gárdase nun obxecto String concatenando caracter a caracter
100  5- Detéctase o final da petición cunha liña en branco
101  6- Envíase a cabeceira HTML
102 */
103
104 void loop() {
105
106   EthernetClient client = server.available();
107   if (client) {
108     Serial.println("Got a client");
109     boolean lineaActualEstaVacia = true;
110     while (client.connected()) {
111       if (client.available() > 0) {
112         char c = client.read();
113         if (peticion.length() < 15) {
114           peticion.concat(c);
115         }
116         if (c == '\n' && lineaActualEstaVacia) {
117           client.println("HTTP/1.1 200 OK");
118           client.println("Content-Type: text/html");
119           client.println();
120           client.println("<!DOCTYPE HTML>");
121           client.println("<html>");
122           client.println("<head>");
123           client.println("<title> Servidor Arduino </title>");
124           client.println("</head>");
125           client.println("<h1> <center> <font face=ARIAL> DATOS ARDUINO <font> <
    center> </h1>");
126
127          car_ASCII[0] = peticion.charAt(12);
128          car_ASCII[1] = peticion.charAt(13);
129          car_ASCII[2] = peticion.charAt(14);
130
131 //Os obxectos String traballan con caracteres ASCII que ocupan un byte cada un.
132
133
```

```
134         if (peticion.charAt(11) == 61) //Comprobamos que o que chega ó buffer é un
    =, porque isto significa que se pulsou o botón enviar
135         {
136           if (peticion.charAt(12) < 48 || peticion.charAt(12) > 59 ) //
    Comprobamos se o recibido é unha letra ou un número
137           {
138             client.print("<br/>"); //Se é unha letra amosase unha mensaxe de erro
139             client.println("Non se admiten letras");
140             client.print("<br/>");
141           }
142             //Funcion que converte secuencias numéricas de caracteres ASCII en
    valores binarios
143           else
144           {
145             if (car_ASCII[1] == 32)
146              {
147                car_dec = car_ASCII[0]-48;
148              }
149             else if (car_ASCII[2] == 32)
150              {
151                car_dec = ((car_ASCII[0]-48)*10 + (car_ASCII[1]-48));
152              }
153             else
154              {
155                car_dec = ((car_ASCII[0]-48)*100+(car_ASCII[1]-48)*10+(car_ASCII
    [2]-48));
156              }
157            }
158
159         if (car_dec < 1 || car_dec > 255) //Comprobamos que o número esté entre
    1 e 255
160         {
161           client.print("<br/>");
162           client.println("Os modulos deben estar numerados entre 1 e 255"); //Se
     non está amósase outra mensaxe de erro
163           client.print("<br/>");
164         }
165         else //Se o número é correcto envíase unha petición a dirección
166         {
167
168            if (peticion.indexOf("XBee+=") > 0 ) //Indicativo de que se
    introduciu un valor
169            {
170              payload[0] = 'S';
171              dir = car_dec;
172              envio();
173              lectura();
174              if (ack)        //Comprobase que o mensaxe chegou ao destinatario
175              {
176                //Información da páxina HTML
```

```
                    client.println("ID: ");
                    client.print(id_h, HEX);
                    client.println(id_l, HEX);
                    dato = (data_h << 8) + data_l;
                    client.print("<br/>");
                    client.println("Lectura sensor: ");
                    client.println(dato);
                  }
                 else
                 {
                    client.println("Modulo non operativo");
                 }
                }
              }
            }

            //Botones de control
            client.println("<br/>");
            client.println("<form>");
            client.println("<input type='text' name='XBee '/>");
            client.println("<button type='submit' value='0'>Enviar </button>");
            client.println("</form>");
            client.println("</html>");
            break;
            }
            //Comprobación de que rematou a petición
            if (c == '\n') {
              lineaActualEstaVacia = true;
            } else if (c != '\r' ) {
              lineaActualEstaVacia = false;
            }
          }
        }
        peticion="";
        delay(500);
        client.stop();
        }
}
```

Código 10.2: Arduino NANO - Escravo da rede

```
//Arduino NANO – Escravo da rede

//Biblioecas de funcións XBee e display LCD
#include <XBee.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

```

```
LiquidCrystal_I2C lcd(0x27,16,2);  //Indicación de que o display é de 16 columnas e
    2 filas

XBee xbee = XBee();
XBeeResponse respuesta = XBeeResponse();

uint8_t data = 0;
uint8_t payload[] = {0,0,0,0};
uint8_t my[] = {'M','Y'}; //Comando AT que se solicitará

uint8_t dato_l = 0;
uint8_t dato_h = 0;

Rx16Response rx16 = Rx16Response();
Tx16Request tx = Tx16Request (0x0002, payload, sizeof(payload));
TxStatusResponse txStatus = TxStatusResponse();

AtCommandRequest atRequest = AtCommandRequest(my);

AtCommandResponse atResponse = AtCommandResponse();

int pot = 0;
int contador = 0;
boolean ack = false;
int dato = 0;

/* Inicialización:
   1- Encendido LCD
   2- Brillo de fondo do LCD
   3- Arranque do porto serie
   4- Asignación do porto serie ao módulo XBee
   5- Petición de identificador de nodo
*/

void setup() {
  lcd.init();
  lcd.backlight();
  Serial.begin(9600);
  xbee.setSerial(Serial);
  sendAtCommand();
}

//Función de lectura de datos RF
void lectura() {
  xbee.readPacket();
  if (xbee.getResponse().isAvailable()) {
    if (xbee.getResponse().getApiId() == RX_16_RESPONSE) {
      xbee.getResponse().getRx16Response(rx16);
      data = rx16.getData(0);
      }
```

```
  }
}

//Función de envío de datos RF
void envio() {
  xbee.send(tx);
  if (xbee.readPacket(5000)) {
      if (xbee.getResponse().getApiId() == TX_STATUS_RESPONSE) {
        xbee.getResponse().getZBTxStatusResponse (txStatus);
        if (txStatus.getStatus() == SUCCESS) {
          ack = true;
          } else {
          ack = false;
          }
        }
    }
}

/* Bucle principal
   1- Lectura de datos RF
   2- Compróbase que chega un 'S' para enviar os datos
   3- Lectura da entrada analóxica correspondente
   4- Dado que os datos se envían en 8 bits e a lectura analóxica é de 10 bits
    divídese en dous
   5- Envíase a información
*/

void loop() {
  data = 0;
  lectura();

if (data ==  'S') {
  contador = contador + 1;
  pot = analogRead(0);
  payload[0] = (pot >> 8 & 0xFF);
  payload[1] = (pot & 0xFF);
  payload[2]= dato_l;
  payload[3]= dato_h;
  lcd.setCursor(0,0);
  lcd.print(payload[3], HEX);
  lcd.print(payload[2], HEX);
  lcd.setCursor(0,1);
  lcd.print(contador);
  lcd.setCursor(3,0);
  lcd.print(payload[0], BIN);
  lcd.setCursor(3,1);
  lcd.print(payload[1], BIN);
  lcd.setCursor(12,0);
  lcd.print(pot);
  lcd.print("   ");
```

```
106   envio();
107 }
108 }
109
110 //Función de lectura de parámetros mediante comandos AT
111 void sendAtCommand() {
112   xbee.send(atRequest);
113
114   if (xbee.readPacket(100))
115   {
116     if (xbee.getResponse().getApiId() == AT_COMMAND_RESPONSE)
117     {
118       xbee.getResponse().getAtCommandResponse(atResponse);
119       if (atResponse.isOk()) {
120         if (atResponse.getValueLength() > 0) {
121           dato_h = (atResponse.getValue()[0]);
122           dato_l = (atResponse.getValue()[1]);
123
124         }
125       }
126   }
127 }
128 }
```

11 Cálculos

A continuación preséntanse os cálculos que foron precisos para o correcto desenrolo deste deseño, así como a súa xustificación.

11.1. Divisor de tensión da batería

As placas escravas da rede precisan dunha alimentación externa, que se fará a través dunha batería que ten que superar o limiar dos 6 voltios para que os circuítos se alimenten correctamente. Por outra banda temos como requisito de deseño que se poda facer un seguimento do estado da batería destes módulos, que se faría a través dunha entrada analóxica que mediría o nivel de tensión que está a dar ese elemento. As entradas analóxicas da placa de programación non soportan niveis de tensión por riba dos 5 voltios, polo que se precisará dun divisor de tensión.

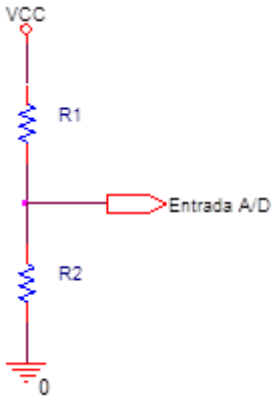


Figura 11.1.0.1 – Divisor de tensión

$$EntradaA/D = Vcc\,\frac{R_2}{R_1 + R_2}$$

En función do voltaxe de entrada escolleranse as resistencias correspondentes. Neste caso a alimentación que se considera máis axeitada é a que proporcionan as baterías comerciais de 9 voltios, polo que se podería implementar un divisor de tensión que divida pola metade. Isto faríase con dúas resistencias de valor igual, que preferiblemente debería ter un valor elevado, para impedir que distorsionen a medida final.

11.2. Equivalencia entre os datos obtidos no A/D e o nivel de batería

O valor que sae dunha medida analóxica das placas de programación Arduino é un valor entre 0 e 1024 para tensións entre 0 e 5 voltios. Para coñecer a equivalencia entre o valor que dará a función de lectura da batería e o nivel real precísase dunha conversión.

En primeiro lugar, debido a que a tensión que verá a entrada da placa variará entre 0 e 4,5 voltios, en lugar dos 5 habituais, calcúlase o novo fondo de escala da medición:

$$V_{max\ de\ entrada}\frac{2^{10}-1}{V_{ref}} = Bits\ FS$$

$$4,5\ \frac{2^{10}-1}{5} = 921$$

Con este dato xa se pode facer a conversión real

$$Lectura\ A/D\frac{V_{ref}}{Bits\ FS} \bullet FR = Tension\ da\ bateria$$

$$Lectura\ A/D\frac{5}{921} \bullet 2 = Tension\ da\ bateria$$

11.3. División de valores de 10 bits para enviarse como paquetes de 8 bits

Os módulos XBee traballan con paquetes de datos de 8 bits, polo que as medidas que superen este limiar deberán dividirse para seren enviadas. Collendo como exemplo as medidas do A/D que son de 10 bits o proceso sería o seguinte:

- **Byte alto:** dato desprazado a dereita 10 posicións e multiplicado por 255.
- **Byte baixo:** dato multiplicado por 255

Exemplo: Co número 455 (01 1100 0111)

Cálculo de byte alto

1. Desprazamento de 8 posicións a dereita

$$01\ 1100\ 0111 \rightarrow 00\ 0000\ 0001$$

2. Multiplicado por 255

$$00\ 0000\ 0001 \bullet 1111\ 1111 = 00\ 0000\ 0001$$

Byte alto: 0000 0001

Cálculo de byte baixo

1. Dato multiplicado por 255

$$01\ 1100\ 0111 \bullet 1111\ 1111 = 01\ 1100\ 0111$$

Byte baixo = 01 1100 0111

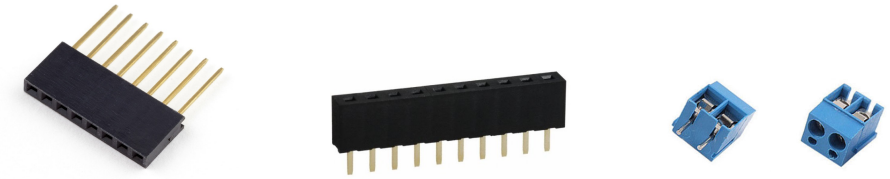
12 Outros anexos

12.1. Placa escravo de rede

Por comodidade e practicidade, durante o deseño decídese construír un elemento que inclúa no mesmo espazo a placa Arduino NANO e o módulo XBee coas conexións precisar para a súa funcionalidade.

En primeiro lugar deben escollerse os zócalos sobre os que irán montadas as dúas partes móbiles do elemento. O paso dos contactos do Arduino NANO segue a métrica inglesa e é de unha décima de pulgada, mentres que o paso do módulo XBee segue a métrica internacional e é de 2 mm.

O resto de conectores da placa serán da medida e formato que se desexe. Neste deseño utilizáronse conectores de parafuso para PCB, xa que permiten unha conexión fácil e forte dos distintos elementos.



(a) Conector de 0,1” (b) Conector de 2 mm (c) Conector de parafuso

Figura 12.1.0.1 – Tipos de conectores

O deseño contempla dous bloques (entradas analóxicas e entradas/ saídas dixitais) con 15 destes contactos cada un distribuídos da seguinte forma:

- Entrada analóxica - Entrada/ Saída dixital
- GND
- $+5$ V

Outro bloque con 4 contactos para a comunicación I^2C cos terminais SDA, SCL, $+5$V e GND e outro de dous coa entrada de alimentación ao circuíto.

No esquemático de deseño inclúese un conxunto de bobina e condensador por cada entrada a placa Arduino. Isto faise como medida de seguridade para protexer os contactos de interferencias, pero poderían obviarse á hora de realizar o montaxe final.

Para o módulo XBee instálase un regulador de tensión LM1117 (ou similar) que aporte unha saída estable de 3.3V e con intensidade suficiente para a transmisión de datos. A este regulador acompáñano os correspondentes condensadores de filtrado de entrada e de saída. Acompañando o módulo de radio tamén están un transistor que une o RESET da placa Arduino ao procesador do XBee para sincronizalos, e un LED que indica que o módulo está asociado mediante radio frecuencia a outro.

Por último incorpórase un interruptor que conecte e desconecte as liñas do porto serie asíncrono da placa Arduino coas do módulos de radio frecuencia. Deste xeito permútase o uso deste medio entre a conexión USB e a conexión XBee.

En caso de querer utilizar este conxunto de elementos no exterior deberíase colocar dentro dunha caixa illante cun grao de protección IP65 (estanco ao po e resistente a chorros de auga).

12.2. Aplicacións

Como se explica anteriormente os código presentes neste documento unicamente contemplan as accións base da comunicación e unha proba sinxela do funcionamento. Para poder extrapolar estes códigos a un uso real precisarían de modificacións. A continuación explicaranse algunhas aplicacións reais deste deseño, xunto coas posibles modificacións do código para aplicalas.

Rede de telemedida de contadores de auga

Actualmente o abastecemento de auga potable nos fogares faise mediante empresas municipais ou concesionarias polos concellos. Isto fai que a contabilización do consumo desta non estea tan estandarizado como o doutros elementos como son a electricidade ou os combustibles. Ademais cando se trata de núcleos rurais ou afastados do centro do municipio, a medida destes consumos faise máis complicada debido ao que os contadores en ocasións están no interior das casas ou simplemente que a persoa que ten que anotar a medición non sabe nin sequera ónde está o contador. O deseño desta documentación permitiría crear unha rede sen fíos entre casas para realizar as medicións correspondentes.

Para iso sería preciso construír unha placa escravo para cada contador cun medidor de fluxo de auga nunha das entradas analóxicas. Cada unha destas formaría parte dunha rede que iría asociada a un coordinador, que controlaría (en caso de que o alcance o permita) ata 255 destes elementos. Estes coordinadores poderían ser o punto final da rede ou poderían formar parte a súa vez dunha rede maior asociada a un coordinador xeral que estaría nun lugar centralizado e que controlara todos os dispositivos. Debido a que as medidas de consumo de auga non se fan de xeito diario, os módulos escravos poderían estar en posición de baixo consumo a maior parte do día e espertarse unha vez cada certo tempo para tomar a medida de almacenala ata que o coordinador lla pedise.

Modificacións do código

No código dos escravos habería que incluír un array das posicións que se queira para almacenar as medicións que fai o dispositivo. Estas poden gardarse na memoria EEPROM incluída na placa, xa que ocuparán un volume reducido. Tamén habería que incluír a lectura dunha entrada analóxica, onde iría o divisor de tensión da batería. Para diferenciar entre cando se quere mostrar a lectura do sensor ou a lectura das baterías habería que crear un novo código de petición.

No código do mestre ou coordinador de rede, habería que cambiar a definición da páxina HTML para que cree un novo botón coa función precisa.

Domótica

A domótica é a rama da automatización aplicada a vivendas. Consiste en automatizar o control de certos procesos cotiáns que se levan a cabo nos fogares.

Con este deseño sería posible crear unha rede domótica sen necesidade dunha instalación cableada xa que a comunicación funcionaría vía radio. Poñendo como exemplo un fogar normal os controis a efectuar serían:

- Subida e baixada de persianas
- Encendido e apagado de luces
- Encendido e apagado de calefacción
- Peche automático de portas

Cun módulo de saída a relé conectado entre as alimentacións destes elementos poderíase controlar dende unha páxina web a actuación sobre estes elementos.

Instalaríase un escravo cun módulo de relé en cada un dos actuadores que realizan estas accións:

- Subida e baixada de persianas $\rightarrow$ driver de control de motores (sentido de xiro, arranque e parada)
- Encendido e apagado de luces $\rightarrow$ interruptor
- Encendido e apagado de calefacción $\rightarrow$ interruptor ou termostato
- Peche automático de portas $\rightarrow$ accionamento electro mecánico

Os módulos de relé funcionan conmutando unha saída de tensión de rede (220V 50Hz) cun sinal de entrada en lóxica TTL. Deste xeito, a partir das saídas dixitais da placa Arduino poderían conmutarse cargas de alterna.

Modificacións de código

Engadir as instrucións de activación dos contactos dixitais correspondentes ás saídas de relé nas placas escravo e crear o entorno HTML correspondente (botóns de acción) no mestre.

TÍTULO: **DESEÑO DUNHA REDE SEN FÍOS DE TELEMEDIDA**

PLANOS

PETICIONARIO: **ESCOLA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

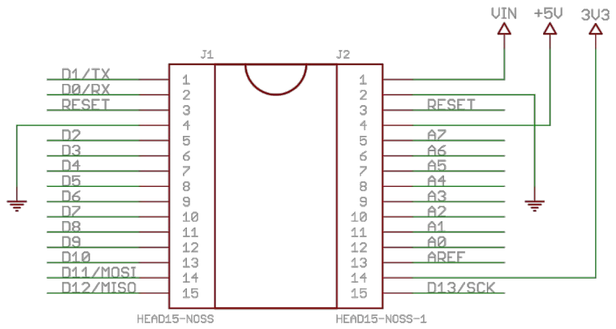
DATA: FEBREIRO DE 2016

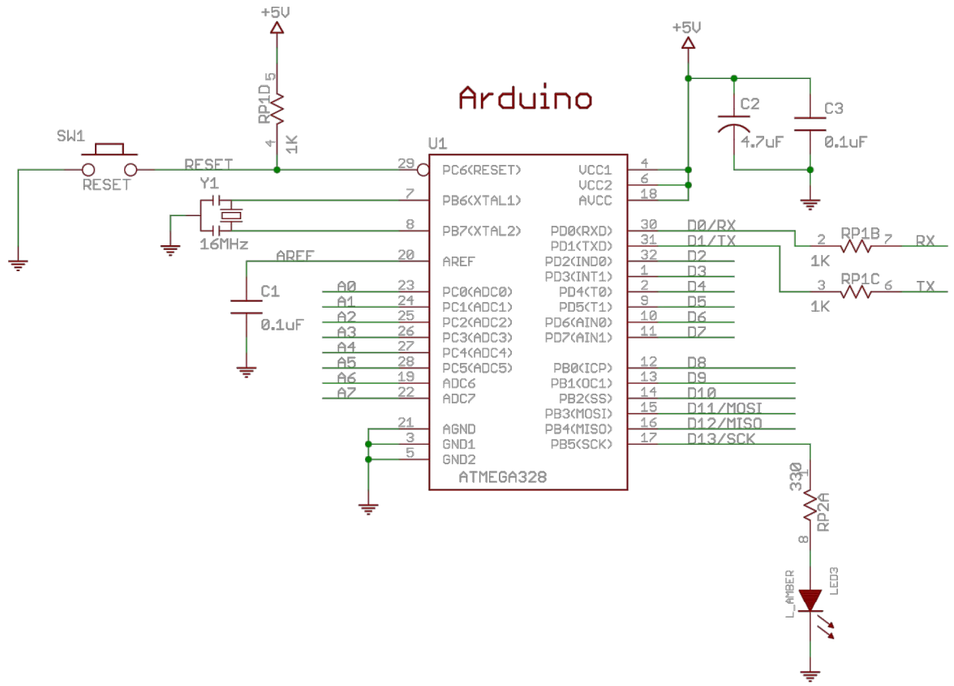
AUTOR: O ALUMNO

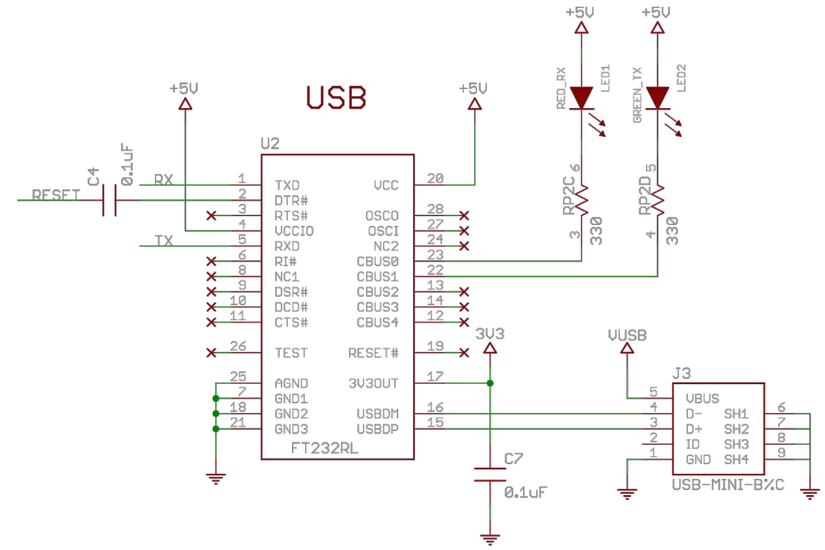
Fdo.: VÍCTOR MANUEL ANIDOS BLANCO

Índice de planos

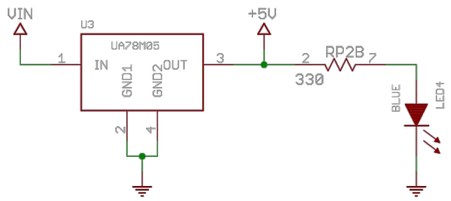
Arduino Nano





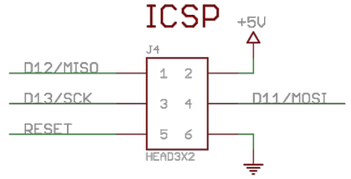


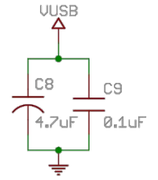
+5V REG



+5V AUTO SELECTOR

ICSP





NOT USED

UNIVERSIDADE DA CORUÑA ESCOLA UNIVERSITARIA POLITÉCNICA

GRAO EN ENXEÑARÍA ELECTRÓNICA INDUSTRIAL E AUTOMÁTICA

TFG Nº: 770G01V01

TÍTULO DO TFG:

DESEÑO DUNHA REDE SEN FÍOS DE TELEMEDIDA

TÍTULO DO PLANO:

ESQUEMÁTICO ARDUINO NANO

DATA: FEBREIRO 2015

ESCALA: 1:1

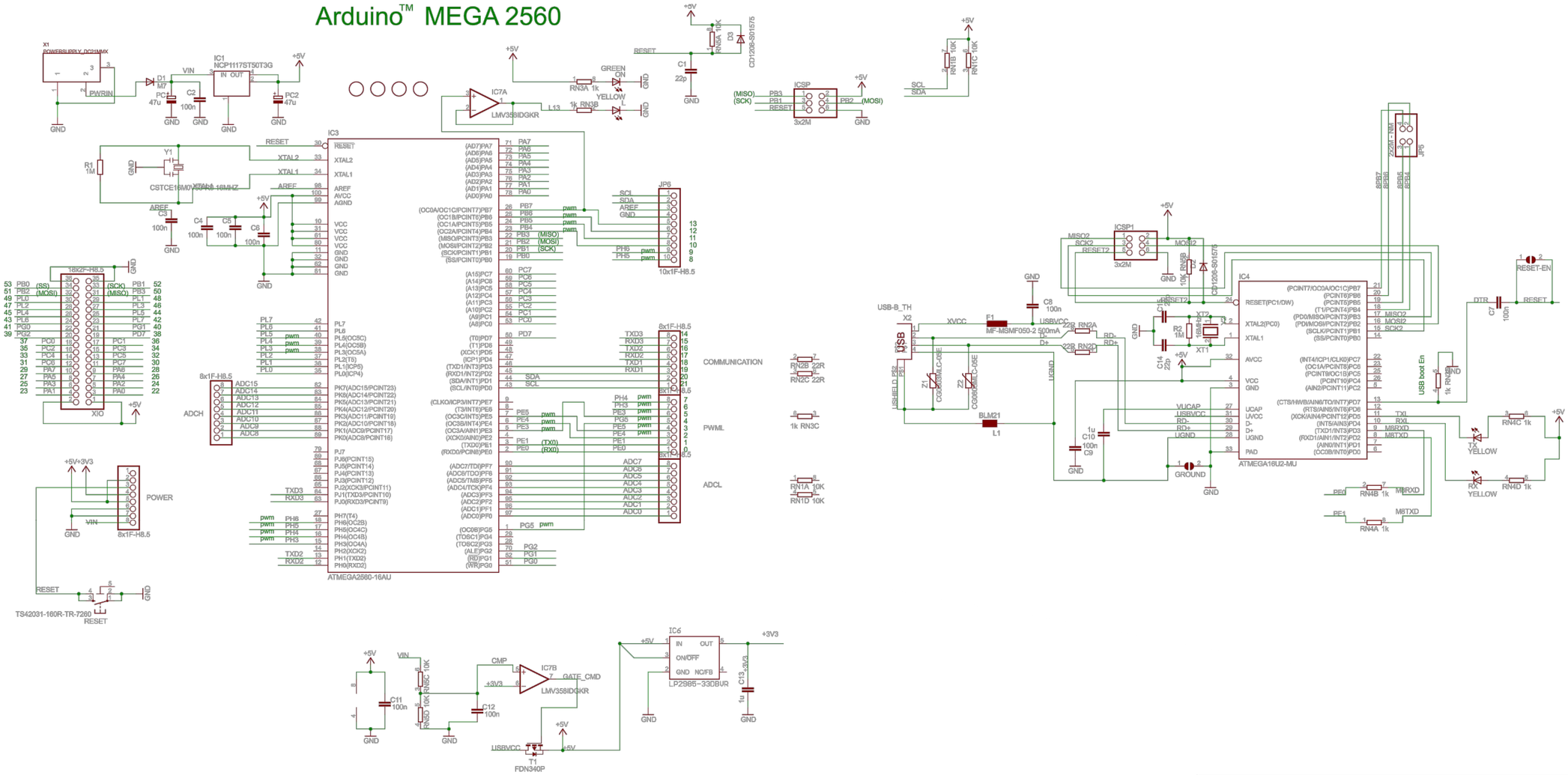
AUTOR:

VÍCTOR MANUEL ANIDOS BLANCO

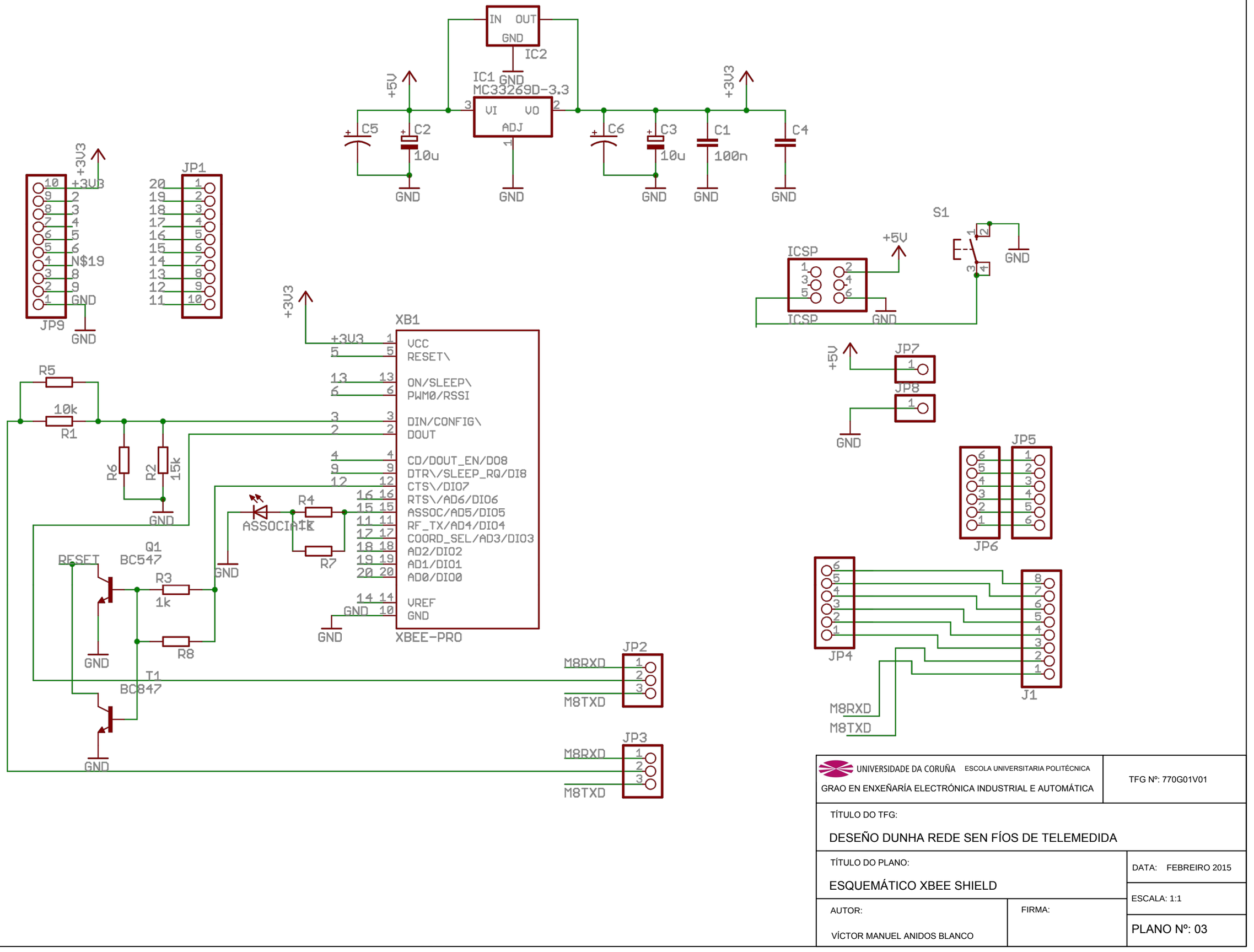
FIRMA:

PLANO Nº: 01

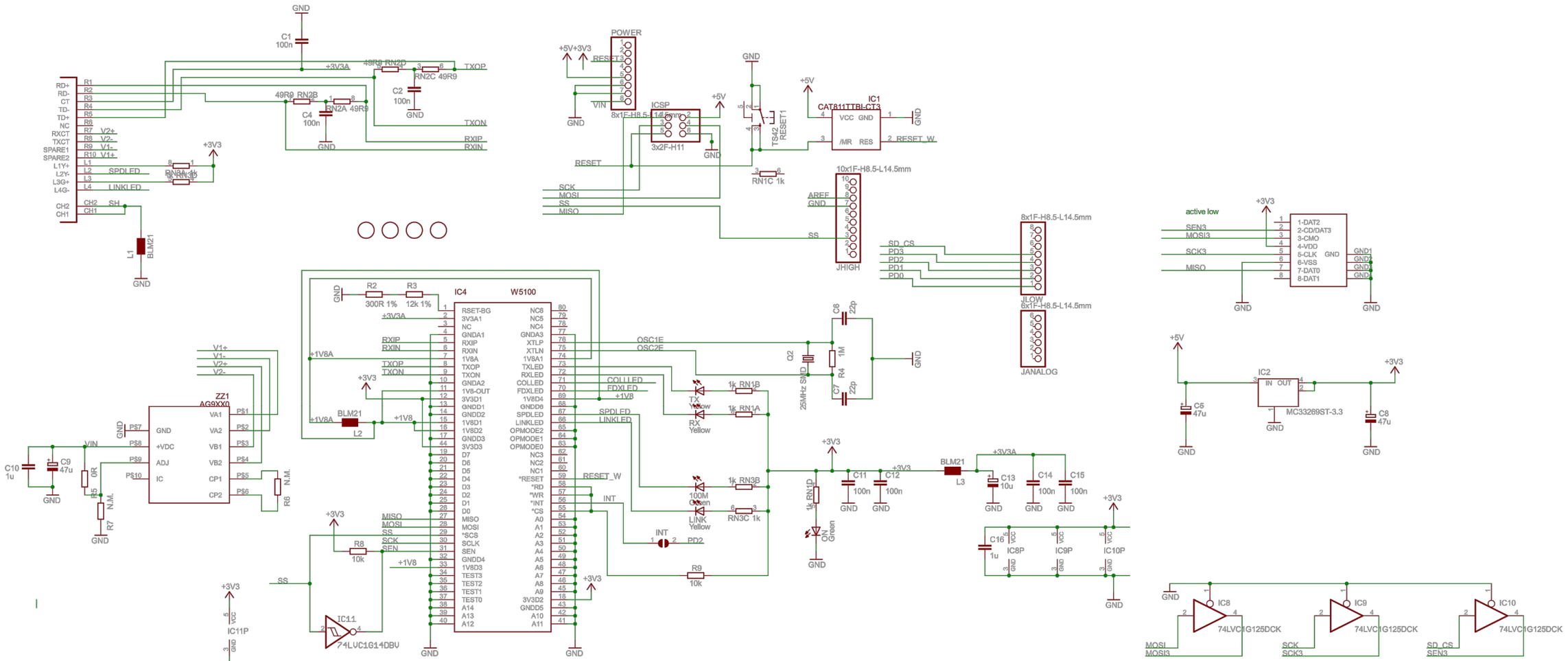
Arduino™ MEGA 2560



UNIVERSIDADE DA CORUÑA ESCOLA UNIVERSITARIA POLITÉCNICA GRAO EN ENXEÑARÍA ELECTRÓNICA INDUSTRIAL E AUTOMÁTICA	TFG Nº: 770G01V01
TÍTULO DO TFG: DESEÑO DUNHA REDE SEN FÍOS DE TELEMEDIDA	
TÍTULO DO PLANO: ESQUEMÁTICO ARDUINO MEGA	DATA: FEBREIRO 2015
	ESCALA: 1:1
AUTOR: VÍCTOR MANUEL ANIDOS BLANCO	FIRMA:
	PLANO Nº: 02

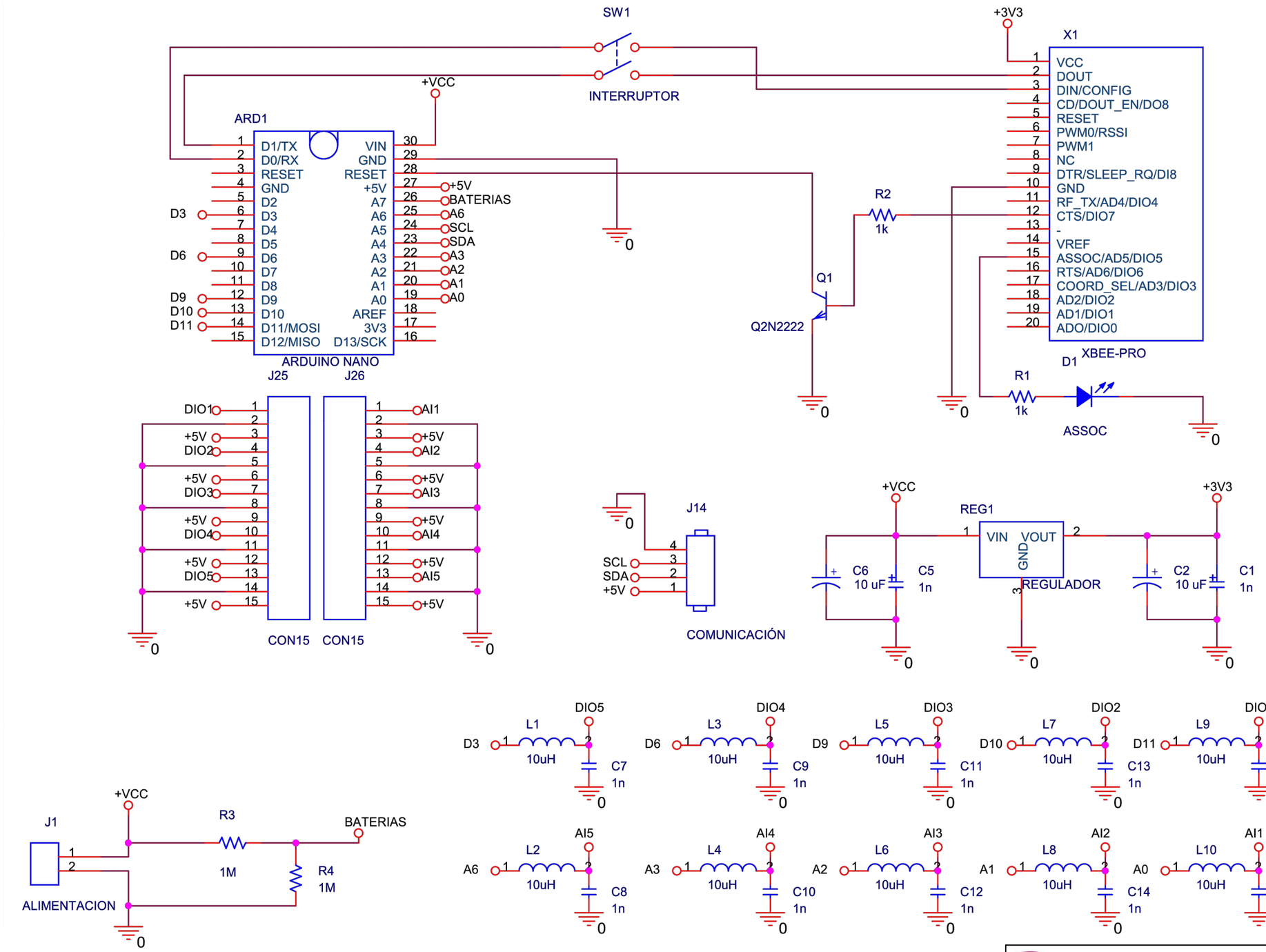


UNIVERSIDADE DA CORUÑA ESCOLA UNIVERSITARIA POLITÉCNICA GRAO EN ENXEÑARÍA ELECTRÓNICA INDUSTRIAL E AUTOMÁTICA	TFG Nº: 770G01V01
TÍTULO DO TFG: DESEÑO DUNHA REDE SEN FÍOS DE TELEMEDIDA	
TÍTULO DO PLANO: ESQUEMÁTICO XBEE SHIELD	DATA: FEBREIRO 2015 ESCALA: 1:1
AUTOR: VÍCTOR MANUEL ANIDOS BLANCO	FIRMA: PLANO Nº: 03



ARDUINO ETH SHIELD 06 - Rev 03

UNIVERSIDADE DA CORUÑA ESCOLA UNIVERSITARIA POLITÉCNICA GRAO EN ENXEÑARÍA ELECTRÓNICA INDUSTRIAL E AUTOMÁTICA	TFG Nº: 770G01V01
TÍTULO DO TFG: DESEÑO DUNHA REDE SEN FÍOS DE TELEMEDIDA	
TÍTULO DO PLANO: ESQUEMÁTICO ETHERNET SHIELD	DATA: FEBREIRO 2015 ESCALA: 1:1
AUTOR: VÍCTOR MANUEL ANIDOS BLANCO	FIRMA: PLANO Nº: 04



UNIVERSIDADE DA CORUÑA ESCOLA UNIVERSITARIA POLITÉCNICA GRAO EN ENXEÑARÍA ELECTRÓNICA INDUSTRIAL E AUTOMÁTICA	TFG Nº: 770G01V01
TÍTULO DO TFG: DESEÑO DUNHA REDE SEN FÍOS DE TELEMEDIDA	
TÍTULO DO PLANO: PLACA ARDUINO ESCRAVO	DATA: FEBREIRO 2015 ESCALA: 1:1
AUTOR: VÍCTOR MANUEL ANIDOS BLANCO	FIRMA: PLANO Nº: 05

TÍTULO: **DESEÑO DUNHA REDE SEN FÍOS DE TELEMEDIDA**

PREGO DE CONDICIÓNS

PETICIONARIO: **ESCOLA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

DATA: FEBREIRO DE 2016

AUTOR: O ALUMNO

Fdo.: VÍCTOR MANUEL ANIDOS BLANCO

Índice do documento PREGO DE CONDICIÓNS

12.3. Obxecto

O prego de condicións defínese como o documento que especifica as condicións técnico facultativas para a execución de elementos físicos, determinando con carácter xeral as obrigas das partes que interveñen no proceso de execución do presente traballo. O prego de condicións xerais define con un carácter xenérico os aspectos da execución dos traballos.

Este traballo de fin de grao, consiste unicamente no deseño do protocolo de xestión da rede que conformarán os elementos aquí definidos. Polo tanto o produto final que sairá del serán uns códigos de programación destinados a gravar sobre unhas plataformas definidas. Para o desenrolo destes códigos foi preciso material, equipos e montaxe. Eses requirimentos serán sobre os que se aplicará este prego de condicións, pero é preciso remarcar que os códigos saídos desta documentación poderán ser usados da forma que o usuario final crea oportuna sen ter que cinguirse ao material que se menciona aquí.

Este prego de condicións ten por obxecto determinar as condicións mínimas aceptables para a execución do montaxe dos elementos obxecto do deseño. Refírense a instalación dos materiais necesarios na instalación, fixando os niveis técnicos e de calidade esixibles, precisando as intervencións que correspondan.

12.4. Documentación

Integran a execución do deseño, os seguintes documentos relacionados por orde:

1. Prego de condicións particulares
2. Prego xeral de condicións
3. Prezos unitarios
4. Planos
5. A memoria, en canto a definición de materiais e calidades.

12.5. Prego de condicións xerais

O propietario é a Escola Universitaria Politécnica de Ferrol que encomenda estre traballo de fin de grao a Víctor Manuel Anidos Blanco.

Este prego fai referencia á execución material do deseño presente nesta documentación, que consta dunha rede dun mínimo de 1 e un máximo de 255 elementos independentes de medida, conectados vía radio frecuencia a un elemento coordinador que á súa vez terá conexión ethernet e permitirá ser manexado dende unha páxina web. Nel contemplarase a correcta instalación e posta en funcionamento dos equipos.

- **Normativa de aplicación**

  A continuación detallarase a normativa española e europea a ter en conta na execución do deseño:

  - Cadro nacional de atribución de frecuencias (CNAF) recollido na orde IET/797/2013 do 25 de abril.
  - Orde IET/1311/2013 do Ministerio de Enerxía, Industria e Turismo, recollida na disposición 7624 do BOE do 12 de xullo de 2013 pola que se aproba o Regulamento do uso do dominio público radio eléctrico por radio afeccionados.
  - Orde do 25 de xuño de 2008 pola que se establecen as especificacións técnicas dos equipos comerciais de radio afeccionados publicada no BOE número 165 do 11 de xullo de 1998.
  - Directiva 2004/108/CE do Parlamento Europeo e do Consello do 15 de decembro de 2004 relativa á aproximación das lexislacións dos estados membros en materia de Compatibilidade Electromagnética publicado no Diario Oficial da Unión o 31 de decembro de 2004.
  - Directiva 2011/65/UE do Parlamento Europeo e do Consello do 8 de xuño de 2011 sobre restricións á utilización de determinadas sustancias perigosas en aparellos eléctricos ou electrónicos.
  - Real Decreto 110/2015 do 25 de febreiro de 2015 sobre residuos de equipos eléctricos e electrónicos.
  - Directiva 2012/19/UE do Parlamento Europeo e do Consello do 4 de xullo de 2012 sobre residuos de aparellos eléctricos e electrónicos (RAEE).
  - Directiva 93/68/CEE do Consello do 22 de xullo de 1993 pola que se modifican as directivas 89/336/CEE de compatibilidade electromagnética, 91/263/CEE de equipos terminais de telecomunicacións e 73/23/CEE de material eléctrico.

12.6. Prego de condicións técnicas

12.6.1. Especificacións de materiais e equipos

Os materiais precisos para execución deste deseño dependerán do tipo de instalación que se desexe. A diferenza está na forma de construción dos módulos independentes: por un lado existe a opción de utilizar módulos separados para cada función, existentes no mercado, os cales irían unidos mediante cables nunha placa de prototipos. Ou facer unha placa de circuíto impreso onde estarían incluídas todas as funcionalidades requiridas para o deseño.

Optando pola primeira opción a lista de elementos necesarios é a seguinte:

Unidade mestra:

- Arduino MEGA 2560
- XBee shield
- Módulo XBee XB24-AWI-001 con antena de cable
- Ethernet shield
- Router
- Cable UTP 8 categoría 5 con cabezais RJ45

Unidade escrava *(entre 1 e 255)*

- Arduino NANO
- XBee shield
- Módulo XBee XB24-AWI-001 con antena de cable
- Conector tipo broche para pilas de 9 voltios

Optando pola segunda opción:

Unidade mestra: igual

Unidade escrava:

- Arduino NANO
- Módulo XBee XB24-AWI-001 con antena de cable
- Conector tipo broche para pilas de 9 voltios
- Tira de 15 contactos metálicos de paso 0,1”. (x2)
- Tira de 10 contactos metálicos de paso 2 mm (x2).
- Tira de 15 contactos de parafuso. (x2).
- Tira de 4 contactos de parafuso (x1).
- Tira de 2 contactos de parafuso (x1)
- Interruptor de 4 vías e dúas posicións.
- Regulador de tensión de 3,3 voltios.
- Condensadores cerámicos 1 nF (x2)
- Condensadores electrolíticos de 10 microF

- Resistencia de 1/4 W de 1k (x2)
- Resistencia de 1/4 W de 1 M (x2)
- Transistor NPN Q2N2222
- LED
- Placa virxe de fibra de vidro (100x160 mm)

Todos e cada un dos elementos precisarán ter un certificado conforme cumpren as directivas europeas de calidade mencionadas no prego de condicións xerais.

En canto aos equipos, para calquera das dúas opcións será preciso un PC con conexión á mesma rede ethernet onde está conectada a placa mestra, cun navegador web que soporte o linguaxe no que estea escrito o programa.

No caso da construción da placa escravo os requirimentos de equipos aumentan. Para levala a cabo precisarase un equipamento de fabricación de placas de circuíto impreso, que pode ser tanto manual como automatizado, ou ordenado externamente a unha terceira entidade.

12.6.2. Especificacións de execución

En canto á construción da placa de circuíto impreso que albergará os módulos escravos, non compete á redacción deste prego a indicación de montaxe e instalación, xa que se contempla como un engadido extra á base de fundamento e xustificación do deseño. Polo que os criterios de montaxe, posta en funcionamento e probas de uso son responsabilidade do usuario final ou da entidade á que se encargue o traballo.

Na parte que compete a este prego, as especificacións de execución baséanse na modificación dos códigos de programación. Considérase que as únicas modificacións precisas son as referentes a especificacións individuais concretas de cada módulo, e que serán modificacións ao funcionamento das entradas e saídas tanto analóxicas como dixitais do soporte de programación Arduino. A parte que relacionada ó fluxo de datos entre as estacións de radio e a tarxeta de ethernet considérase suficientemente desenrolada ao usos que recibirán estes equipos e polo tanto non susceptibles de modificacións. En caso imperioso de que se necesite un cambio nesta parte, deberá levarse a cabo a través dunha persoa coa base técnica suficiente e amplamente coñecedora dos equipos cos que se está a traballar.

O montaxe final do deseño deberá ser realizado por persoas coa cualificación suficiente (graduado en Enxeñaría Industria/ Enxeñeiro técnico) co fin de asegurarse que todas e cada unha das partes están conforme o redactado nesta documentación, e que as que non están podas ser modificadas con coñecemento de causa.

Considérase imprescindible a proba de funcionamento de cada unha das partes que conforman o deseño de xeito individual para asegurarse de que o montaxe final funcionará nas condicións axeitadas. Calquera modificación software será primeiro probada en módulos auxiliares e logo introducida nos módulos finais.

Será responsabilidade final do usuario a comprobación periódica das fontes de enerxía dos módulos independentes, facéndose cargo da súa reparación ou substitución en caso de ser necesario.

TÍTULO: **DESEÑO DUNHA REDE SEN FÍOS DE TELEMEDIDA**

ESTADO DE MEDICIÓNS

PETICIONARIO: **ESCOLA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

DATA: FEBREIRO DE 2016

AUTOR: O ALUMNO

Fdo.: VÍCTOR MANUEL ANIDOS BLANCO

Índice do documento ESTADO DE MEDICIÓNS

O documento Estado de medicións ten como misión definir e determinar as unidades de cada partida ou unidade de obra que configuran o produto obxecto do traballo de fin de grao. Dado que o produto saído desta documentación é un software o estado de medicións fará referencia aos materiais precisados para levar a cabo o desenrolo do mesmo, que non teñen por qué coincidir cos que usará o usuario final.

12.7. Placas de desenrolo

	Descrición	Cantidade
Ud.	Arduino MEGA 2560 Rev3 ou equivalente	1
Ud.	Arduino NANO v3.0 ou equivalente	3

12.8. Módulos de radio

	Descrición	Cantidade
Ud.	XBee Series 1 Wire antenna	4
Ud.	XBee shield ou equivalente	4

12.9. Módulo de ethernet

	Descrición	Cantidade
Ud.	Ethernet shield ou equivalente	1
Ud.	Router Huawei OBSERVA BRA14NR ou equivalente	1
Ud.	Cable UTP categoría 5, 1 m	1

12.10. Accesorios de funcionamento e alimentación

	Descrición	Cantidade
Ud.	Breadboard sen soldaduras GS	1
Ud.	Potenciómetro rotatorio 10 kΩ $\pm$20 $\%$	3
Ud.	LCD Dot matrix 16x2	3
Ud.	Módulo Serie I^2C para LCD	3
Ud.	Baterías GP de 9V de cloruro de cinc	3
Ud.	Cable baterías TYP2	3

12.11. Elementos da placa de circuito impreso

	Descrición	Cantidade
Ud.	Conector placa a placa de 8 contactos paso de 0.1”	4
Ud.	Conector placa a placa de 10 contactos paso de 2 mm	2
Ud.	Interruptor DIL 2 vías	1
Ud.	Resistencia de 1 kΩ $\pm 5\,\%$	2
Ud.	Resistencia de 1 MΩ $\pm 5\,\%$	2
Ud.	Transistor P2N2222	1
Ud.	LED verde	1
Ud.	Regulador LM1117	1
Ud.	Condensador de 1 nF	2
Ud.	Condensador de 10 μF	2
Ud.	Contactos de tornillo de 3 entradas	10
Ud.	Contactos de tornillo de 2 entradas	3

12.12. Man de obra

	Descrición	Cantidade
H.	Desenrolo do produto	180
H.	Redacción e documentación do deseño	20
H.	Montaxe sen placa de circuito impreso	1
H.	Montaxe con placa de circuito impreso	30

12.13. Software

	Descrición	Cantidade
Ud.	Arduino IDE 1.6.4	1
Ud.	Cadence OrCAD 10.5	1
Ud.	AutoCAD 2015	1
Ud.	TeXnicCenter	2015
Ud.	X-CTU	1

TÍTULO: **DESEÑO DUNHA REDE SEN FÍOS DE TELEMEDIDA**

ORZAMENTO

PETICIONARIO: **ESCOLA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

DATA: FEBREIRO DE 2016

AUTOR: O ALUMNO

Fdo.: VÍCTOR MANUEL ANIDOS BLANCO

Índice do documento ORZAMENTO

O documento Orzamento ten como misión cuantificar económicamente o deseño da documentación. Dado que o produto é un software e unicamente se poderían cuantificar as horas de man de obra, este orzamento fará referencia aos materiais precisados para levar a cabo o desenrolo do mesmo, que non teñen por qué coincidir cos que usará o usuario final. Farase unha distinción entre as dúas opcións de montaxe contempladas.

13 Prezos unitarios de materiais, man de obra e elementos auxiliares

13.1. Placas de desenrolo

	Descrición	Prezo	
Ud.	Arduino MEGA 2560 Rev3 ou equivalente	35,81	euros
Ud.	Arduino NANO v3.0 ou equivalente	24,01	euros

13.2. Módulos de radio

	Descrición	Prezo	
Ud.	XBee Series 1 Wire antenna	21,16	euros
Ud.	XBee shield ou equivalente	15,90	euros

13.3. Módulo de ethernet

	Descrición	Prezo	
Ud.	Ethernet shield ou equivalente	30,61	euros
Ud.	Router Huawei OBSERVA BRA14NR ou equivalente	99	euros
Ud.	Cable UTP categoría 5, 1 m	1,75	euros

13.4. Accesorios de funcionamento e alimentación

	Descrición	Prezo	
Ud.	Breadboard sen soldaduras GS	29,35	euros
Ud.	Potenciómetro rotatorio 10 kΩ $\pm$20 $\%$	0,40	euros
Ud.	LCD Dot matrix 16x2	8,80	euros
Ud.	Módulo Serie I^2C para LCD	5,48	euros
Ud.	Baterías GP de 9V de cloruro de cinc	1,73	euros
Ud.	Cable baterías TYP2	7,74	euros

13.5. Elementos da placa de circuito impreso

	Descrición	Prezo	
Ud.	Conector placa a placa de 8 contactos paso de 0.1”	0,81	euros
Ud.	Conector placa a placa de 10 contactos paso de 2 mm	1,87	euros
Ud.	Interruptor DIL 2 vías	1,03	euros
Ud.	Resistencia de 1 kΩ $\pm 5\,\%$	0,05	euros
Ud.	Resistencia de 1 MΩ $\pm 5\,\%$	0,05	euros
Ud.	Transistor P2N2222	0,19	euros
Ud.	LED verde	0,06	euros
Ud.	Regulador LM1117	1,23	euros
Ud.	Condensador de 1 nF	0,12	euros
Ud.	Condensador de 10 μF	0,09	euros
Ud.	Contactos de tornillo de 3 entradas	0,41	euros
Ud.	Contactos de tornillo de 2 entradas	0,38	euros

13.6. Man de obra

	Descrición	Prezo	
H.	Desenrolo do produto	15	euros
H.	Redacción e documentación do deseño	10	euros
H.	Montaxe sen placa de circuito impreso	15	euros
H.	Montaxe con placa de circuito impreso	15	euros

13.7. Software

	Descrición	Prezo	
Ud.	Arduino IDE 1.6.4	0	euros
Ud.	Cadence OrCAD 10.5	0	euros
Ud.	AutoCAD 2015	0	euros
Ud.	TeXnicCenter	0	euros
Ud.	X-CTU	0	euros

14 Prezos unitarios das unidades de funcionamento

14.1. Unidade mestra

	Descrición	Cantidade	Prezo unitario		Importe	
Ud.	Arduino MEGA 2560 Rev3 ou equivalente	1	35,81	euros	35,81	euros
Ud.	XBee Series 1 Wire antenna	1	21,16	euros	21,16	euros
Ud.	XBee shield ou equivalente	1	15,90	euros	15,90	euros
Ud.	Ethernet shield ou equivalente	1	30,61	euros	30,61	euros
Ud.	Router Huawei OBSERVA BRA14NR ou equivalente	1	99	euros	99	euros
Ud.	Cable UTP categoría 5, 1 m	1	1,75	euros	1,75	euros
			Total:		204,23	euros

14.2. Unidade escrava (sen placa de circuito impreso)

	Descrición	Cantidade	Prezo unitario		Importe	
Ud.	Arduino NANO v3.0 ou equivalente	1	24,01	euros	24,01	euros
Ud.	XBee Series 1 Wire antenna	1	21,16	euros	21,16	euros
Ud.	XBee shield ou equivalente	1	15,90	euros	15,90	euros
			Total:		61,07	euros

14.3. Unidade escrava (con placa de circuito impreso)

	Descrición	Cantidade	Prezo unitario		Importe	
Ud.	Arduino NANO v3.0 ou equivalente	1	24,01	euros	24,01	euros
Ud.	XBee Series 1 Wire antenna	1	21,16	euros	21,16	euros
Ud.	Conector placa a placa de 8 contactos paso de 0.1”	2	0,81	euros	1,62	euros
Ud.	Conector placa a placa de 10 contactos paso de 2 mm	2	1,87	euros	3,74	euros
Ud.	Interruptor DIL 2 vías	1	1,03	euros	1,03	euros
Ud.	Resistencia de 1 kΩ $\pm 5\,\%$	2	0,05	euros	0,10	euros
Ud.	Resistencia de 1 MΩ $\pm 5\,\%$	2	0,05	euros	0,10	euros
Ud.	Transistor P2N2222	1	0,19	euros	0,19	euros
Ud.	LED verde	1	0,06	euros	0,06	euros
Ud.	Regulador LM1117	1	1,23	euros	1,23	euros
Ud.	Condensador de 1 nF	2	0,12	euros	0,24	euros
Ud.	Condensador de 10 μF	2	0,09	euros	0,18	euros
Ud.	Contactos de tornillo de 3 entradas	10	0,41	euros	4,10	euros
Ud.	Contactos de tornillo de 2 entradas	3	0,38	euros	1,14	euros
			Total:		58,09	euros

14.4. Accesorios de funcionamento e alimentación

	Descrición	Cantidade	Prezo unitario		Importe	
Ud.	Breadboard sen soldaduras GS	1	29,35	euros	29,35	euros
Ud.	Potenciómetro rotatorio 10 kΩ $\pm 20\,\%$	3	0,40	euros	1,20	euros
Ud.	LCD Dot matrix 16x2	3	8,80	euros	26,4	euros
Ud.	Módulo Serie I^2C para LCD	3	5,48	euros	16,44	euros
Ud.	Baterías GP de 9V de cloruro de cinc	3	1,73	euros	5,19	euros
Ud.	Cable baterías TYP2	3	7,74	euros	23,22	euros
			Total:		101,80	euros

14.5. Man de obra (sen placa de circuito impreso)

	Descrición	Cantidade	Prezo unitario		Importe	
H.	Desenrolo do produto	180	15	euros	2700	euros
H.	Redacción e documentación do deseño	20	10	euros	200	euros
H.	Montaxe sen placa de circuito impreso	1	15	euros	15	euros
			Total:		2915	euros

14.6. Man de obra (con placa de circuito impreso)

	Descrición	Cantidade	Prezo unitario		Importe	
H.	Desenrolo do produto	180	15	euros	2700	euros
H.	Redacción e documentación do deseño	20	10	euros	200	euros
H.	Montaxe con placa de circuito impreso	30	15	euros	450	euros
			Total:		3350	euros

15 Orzamento

15.1. Rede sen placas de circuito impreso

	Descrición	Cantidade	Prezo unitario		Importe	
Ud.	Unidade mestra	1	204,23	euros	204,23	euros
Ud.	Unidade escrava (sen placa de circuito impreso)	3	61,07	euros	183,21	euros
Ud.	Man de obra (sen placa de circuito impreso)	–	2915	euros	2915	euros
Ud.	Accesorios de funcionamento e alimentación	–	101,80	euros	101,80	euros

Presuposto de execución material:	3404,24	euros
$13\,\%$ gastos xerais:	442,55	euros
$6\,\%$ beneficio industrial:	204,25	euros
Presuposto bruto:	4051,04	euros
$21\,\%$ IVE:	850,72	euros
TOTAL:	4901,76	euros

15.2. Rede con placas de circuito impreso

	Descrición	Cantidade	Prezo unitario		Importe	
Ud.	Unidade mestra	1	204,23	euros	204,23	euros
Ud.	Unidade escrava (con placa de circuito impreso)	3	58,09	euros	174,27	euros
Ud.	Man de obra (con placa de circuito impreso)	–	3350	euros	3350	euros
Ud.	Accesorios de funcionamento e alimentación	–	101,80	euros	101,80	euros

Presuposto de execución material:	3830,03	euros
$13\,\%$ gastos xerais:	497,94	euros
$6\,\%$ beneficio industrial:	229,80	euros
Presuposto bruto:	4557,77	euros
$21\,\%$ IVE:	957,14	euros
TOTAL:	5514,91	euros

TÍTULO: **DESEÑO DUNHA REDE SEN FÍOS DE TELEMEDIDA**

ESTUDOS CON ENTIDADE PROPIA

PETICIONARIO: **ESCOLA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

DATA: FEBREIRO DE 2016

AUTOR: O ALUMNO

Fdo.: VÍCTOR MANUEL ANIDOS BLANCO

Índice do documento ESTUDOS CON ENTIDADE PROPIA

15.3. Estudo de alcance dos módulos de radio

Un factor fundamental no desenrolo deste deseño é o alcance que teñen os módulos de radio. Nas follas de características dos fabricantes especifícanse os alcances máximos que as distintas modalidades de elementos teñen tanto en interiores como en exteriores.

As probas realizadas neste estudo fixéronse cos módulos de radio XBee Series 1 que teñen un alcance estimado de 30 metros en interior e de 90 en exterior.

En primeiro lugar faise unha proba no interior dunha vivenda, comprobando que o alcance horizontal é máis que suficiente, e sen importar as divisións interiores que teña que atravesar. Na proba vertical, vese que o alcance se reduce, tendo que aproximar emisor e receptor para conseguir unha comunicación continua. Se ben este problema pode ser derivado de fontes de radiación eléctrica que poden aparecer entre o piso alto e o piso baixo. Con todo, a comunicación é perfectamente posible e suficiente para establecer unha rede no interior dunha vivenda.

A proba exterior faise baixo varios supostos:

- O primeiro suposto faise cunha distancia aproximada de 50 metros, en liña recta e sen ningún obstáculo intermedio. Compróbase que a comunicación é excelente.
- O segundo suposto aumenta a distancia a 100 metros e coas mesmas condicións de visibilidade. O resultado segue a ser o esperado.
- O terceiro suposto faise cunha distancia aínda maior, próxima aos 500 metros. A comunicación comeza a perderse e faise precisa unha repetición dos comandos.
- O último suposto é con obstáculos diante (árbores, edificios, coches) é compróbase que o alcance vese reducido a uns 50 metros aproximadamente.