Sistema domótico inalámbrico con portal WEB.

Sergio Fernández Ferri
Tecnologías de Telecomunicación
Estudios de informática, Multimedia y Telecomunicación
Especialidad Sistemas de Telecomunicación

Sistema Arduino

Pere Tuset Peiró
Antoni Morell Pérez

Junio 2017

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Sistema domótico inalámbrico con portal WEB*
Nombre del autor:	*Sergio Fernández Ferri*
Nombre del consultor/a:	*Antoni Morell Pérez*
Nombre del PRA:	*Pere Tuset Peiró*
Fecha de entrega (mm/aaaa):	*06/2017*
Titulación:	*Tecnologías de Telecomunicación*
Área del Trabajo Final:	*Arduino*
Idioma del trabajo:	*Castellano*
Palabras clave	*Domótica, hardware libre, inalámbrico*

Resumen del Trabajo

La presente memoria describe una manera económica y funcional de realizar un sistema de vivienda inteligente (conocido como sistema domótico), con enlaces a nodos remotos mediante RF y acceso a través de portal WEB, mediante el uso de la conocida plataforma *open hardware* denominada Arduino.

El funcionamiento básico del sistema consiste en un nodo central, con microncontrolador Atmel ATMEGA256, que realizará las funciones de administrador, puerta de enlace a Internet, interface entre el usuario y el sistema a través de páginas WEB, autenticador del usuario mediante comunicación cifrada, control de las comunicaciones con los nodos remotos, y control/administración de sensores directamente conectados a él. Por otro lado, los nodos remotos estarán a la espera de las indicaciones del nodo central o *master*, ejecutando las acciones solicitadas y enviando información de retorno para ser procesada y mostrada por el nodo central.

El enfoque del trabajo se dirige a mostrar la implantación del sistema completo de comunicación, tanto del nodo central con el usuario a través de Ethernet (internet para acceso desde el exterior de la red doméstica) como del nodo central con los nodos remotos a través de RF mediante modulación GFSK (con transceptores NRF24l01 de *Nordic Semocnductor*).

Paralelamente, siguiendo con el desarrollo necesario, se mostrará someramente la comunicación HTTP entre el nodo central y el navegador WEB, el lenguaje HTML, JavaScript, el cifrado AES128 y el protocolo de intercambio de claves *Diffie-Hellman.*

Abstract

The present report describes an economical and functional way to make a smart housing system, with links to remote nodes through RF and access through WEB portal, through the use of the well-known open hardware platform called *Arduino*.

The system's basic operation consists of a central node, with Atmel ATMEGA256 MCU, which will perform the administrator functions, Internet gateway, and user-system interface through WEB pages, user authenticator with encrypted communication, control communications with remote nodes, and control / management of sensors built in him. On the other hand, the remote nodes will be waiting master node indications, performing requested actions and returning information to master node to be processed and to be shown.

The work focus is implementation whole communication system, both the central node to user, through Ethernet (whit internet for out-home access network), and the central node to remote nodes through RF by GFSK modulation (with Nordic Semoconductor transponders NRF241).

Following the necessary development, HTTP communication between the central node and the WEB browser, HTML language, JavaScript, AES128 encryption and the Diffie-Hellman key exchange protocol will also be shown briefly.

Índice de contenidos

Índice de ilustraciones

Índice de tablas

1 Introducción.

La cultura *maker* o movimiento *maker* es un concepto moderno que engloba la creciente tendencia social a la fabricación propia de productos. Aplicado al mundo de la tecnología subvierte la adquisición de productos finales por la creación o elaboración de unos propios gracias a conceptos innovadores como el *open source (*código abierto), *crowfunding* (micromecenazgo), *open hardware,* etc.

Internet de las cosas o *Internet of things* (en adelante IoT) es un concepto que hace referencia a la interconexión digital de las cosas con internet. Este joven concepto viene a indicarnos el nuevo futuro que nos ofrece la incorporación de dispositivos digitales de conexión a internet en las cosas. Podremos administrar dispositivos que estén conectados a internet, trabajar e investigar con la información conjunta que esté accesible, ofrecer nuevos productos y servicios hasta ahora no conocidos.

La domótica ya es un concepto más conocido y maduro que todavía tiene mucho camino por recorrer. Trata de ofrecernos una serie de funcionalidades con el fin de proporcionarnos una vida más confortable a través de la automatización de procesos cotidianos como pueden ser el control de la iluminación, persianas climatización y sistemas de seguridad. Además, no solo es un concepto aplicable a las viviendas sino que también representa una mejora importante para comercios, locales y lugares de trabajo.

El desarrollo de este sistema nos brinda la oportunidad de conocer más a fondo la electrónica de los sistemas a utilizar, programación con un lenguaje de código abierto (descendiente de *processing*), apoyar el movimiento *maker,* favorecer y entender el concepto de *IoT,* y proveernos de un sistema de control y administración de nuestra vivienda de manera económica, eficaz y segura.

1.1. Eliminación de barreras tecnológicas y económicas.

Mediante el impulso de sistemas de código y *hardware* abierto se abre un abanico de posibilidades en el desarrollo de nuevas aplicaciones y productos. Además, la inteligencia e investigación conjunta que brindan los sistemas abiertos permite la reducción de costes a niveles accesibles para la gran mayoría de las personas, lo que

a su vez favorece todavía más el desarrollo tecnológico consecuencia de una inteligencia conjunta formada por miles de usuarios desarrollando nuevas aplicaciones para el sistema.

Estamos frente a una explosión de conocimiento y desarrollo que se encuentra en una fase de maduración temprana. Las personas relacionadas con el mundo de las tecnologías de la información y la comunicación pueden sentir mucho más auténtico y accesible su conocimiento gracias a la posibilidad de explotarlo y compartirlo con facilidad.

1.2. Objetivos principales.

El principal objetivo del trabajo realizado es académico y de aprendizaje personal sobre la comunidad Arduino y sus posibilidades como *hardware* libre. Conseguir un conocimiento de la plataforma, el lenguaje de programación, sus librerías, el funcionamiento y la gran variedad de módulos existentes, entre otros.

Además de los objetivos de aprendizaje y conocimiento de la plataforma, podemos enumerar una serie de objetivos técnicos relacionados directamente con el proyecto seleccionado:

- Configurar el sistema central con Arduino MEGA2560 para realizar las funciones de servidor WEB, interface con el usuario y autenticador.
- Configurar el nodo maestro y los nodos remotos en una red de comunicación mediante radiofrecuencia a 2.4 GHz con los módulos NRF24l01.
- Configurar el sensor de temperatura LM35DZ.
- Configurar el sensor de corriente ACS712.
- Configurar los contactos magnéticos para el control del estado de las puertas.
- Configurar el control de la iluminación mediante relés de doble contacto, conmutadores y el ACS712.
- Implementar los códigos en los microcontroladores del nodo central y nodos remotos para que realicen las funciones y algoritmos necesarios para el funcionamiento estable del sistema.

1.3.Beneficios del proyecto.

Con la elección de este proyecto estamos elaborando un sistema domótico económico y funcional gestionable desde internet sin la necesidad de tener una preinstalación cableada para la comunicación entre módulos gracias al sistema de RF. Además podemos explorar las capacidades de un sistema *open source* y *open hardware* que nos permitirá implementar mayores funcionalidades al proyecto a la vez que nos introducimos en la cultura *maker*.

El sistema propuesto tiene unas grandes posibilidades de extensión y mejora, como podremos y observando, gracias a la capacidad de instalación módulos o placas de expansión (*shields*) y código de una manera económica y eficaz. Existen en el mercado diferentes modelos de microcontroladores con variadas capacidades que podrían dotar de mayor versatilidad al sistema conllevando simplemente un mínimo incremento en la inversión.

Además de los objetivos técnicos del trabajo podemos encontrar un objetivo compartido enfocado a empujar y hacer crecer una inteligencia conjunta al compartir la experiencia en la comunidad *maker*.

Finalmente, mediante un proceso de desarrollo de producto final (no reflejado en este proyecto), a través del desarrollo de placas PCB que contengan los dispositivos conjuntos de cada nodo en carcasas adecuadas de material aislante, se puede conseguir un producto capaz de ser puesto en el mercado pudiendo ser instalado con grandes posibilidades de éxito debido a su reducido coste y a la posibilidad de instalación sin cables.

1.4.Organización de la memoria.

El desarrollo de la presente propuesta comienza con la explicación del estado del arte de la domótica y del sistema Arduino ya que estos conceptos serán la base sobre la que se desarrollará el proyecto.

Seguidamente se presentará el esquema general del sistema sobre una vivienda tipo con una descripción general del sistema y sus funcionalidades, explicando cada

una de ellas y presentando un esquema general del funcionamiento, que nos permitirá empezar a conocer que estamos desarrollando.

Una vez conocidas las funciones del sistema que se pretende desarrollar se realizará un estudio de viabilidad económica del proyecto y la planificación del mismo. Se detallarán los costes del proyecto a través del presupuesto y se realizará una comparación con algunos sistemas similares existentes en el mercado.

A continuación se detallarán las principales características de los MCU's ATMega 2560 y ATmega 328, el módulos Ethernet V2, el NRF24l01 y la comunicación entre ellos mediante el bus SPI, ya que estos serán la base electrónica sobre la que trabajará el sistema.

Seguidamente veremos cómo se pueden implementar las funciones de portal WEB, obtención de temperatura, información de consumo, control de la iluminación y el control de estos de las puertas con cada uno de los dispositivos electrónicos concretos, como el módulo Ethernet, el LM35, el ACS712, relés y contactos magnéticos. Se detallarán esquemas eléctricos y códigos para cada uno de los casos.

Finalmente, antes de los apartados de líneas futuras y conclusiones, se presentarán las instalaciones completas por nodos, mostrando los elementos y esquemas que compondrán cada nodo.

Para finalizar se presentará un apartado de líneas futuras y conclusiones dónde se evidenciarán las posibilidades del sistema central de comunicación desarrollado en la memoria. De una manera económica y sencilla para alguien con nociones se podrán ir añadiendo funcionalidades al sistema.

Se incluyen anexos que ayudarán a comprender mejor el funcionamiento global del sistema así como el funcionamiento por módulos.

También se incluye documentación complementaria con los códigos a implementar en los MCU's a través del IDE Arduino (archivos "Nodo_00_v_25.ino" y "Nodos_remotos_v_8.ino"), especificaciones técnicas del chipset WD5500, del NRF24l01, del sensor de temperatura LM35 y del sensor de corriente ACS712.

1.5. Estado del arte.

Domótica

El término “domótica” (del latín *domus*, casa, e informática) tiene varias acepciones, entre ellas la que da el diccionario de la Real Academia Española, que define domótica como: “*el conjunto de sistemas que automatizan las diferentes instalaciones de una vivienda*”. Con un sistema de vivienda inteligente no solo se consiguen comodidades para los usuarios sino que además se consigue obtener un ahorro energético considerable al reducir el uso inadecuado de las instalaciones que consumen energía.

El origen de la domótica se remonta a la década de los setenta, cuando tras muchas investigaciones aparecieron los primeros dispositivos de automatización de edificio basados en la tecnología X-10. Durante los años siguientes le mercado mostró un creciente interés por estos novedosos sistemas tratando de encontrar la casa ideal. Los primeros sistemas comerciales fueron instalados, sobre todo, en Estados Unidos y se limitaban a la regulación de la temperatura ambiente de los edificios de oficina y poco más. Más tarde, tras el auge de los PC (*Personal Computer*), a finales de la década de los 80 y principios de los 90, se empezaros a incorporar en estos edificios los SCE (Sistemas de Cableado Estructurado) para facilitar la conexión de todo tipo de terminales y periféricos entre sí, utilizando un cableado estándar y tomas repartidas por todo el edificio. Este cableado estructurado permitió, no solo transportar datos, sino que también sirvió para conectar sistemas de seguridad y control, por lo que a esos edificios se les comenzó a llamar edificios inteligentes.

El avance tecnológico constante, los productos *open hardware* y *open source*, la proliferación de nuevas empresas y la demanda por parte del público han hecho que los precios de los sistemas domóticos empiecen a reducirse. Sin embargo, actualmente el número de viviendas domotizadas es todavía relativamente bajo respecto al total de viviendas, pero el interés en su adopción y su precio cada vez más asequible está haciendo crecer el número de instalaciones domóticas en funcionamiento.

Existen una gran cantidad de ventajas y confort que nos ofrece disponer de un sistema de vivienda inteligente o domótico. Tal y como podemos ver en la guía técnica de aplicación (GUÍA-BT-51) sobre “Instalaciones de sistemas de automatización, gestión técnica de la energía y seguridad para viviendas y edificios” del R.E.B.T.

(Reglamente Electrotécnico de Baja Tensión) (R.D. 842/2002, Guía-BT-51), en su página 2ª, indica:

> …los sistemas domóticos realizan el control integrado de múltiples elementos de una instalación con los fines principales de:

- Aumentar el confort, mediante la automatización de elementos de una instalación.
- La gestión técnica de la energía, por ejemplo para el ahorro o eficiencia energética.
- Garantizar la seguridad de las personas, los animales o los bienes.
- Permitir la comunicación del sistema con redes de telecomunicaciones externas.

En la Ilustración 1, obtenida también de la guía técnica de aplicación (GUÍA-BT-51), se observan las diferentes instalaciones que conviven con un sistema domótico o de vivienda inteligente.

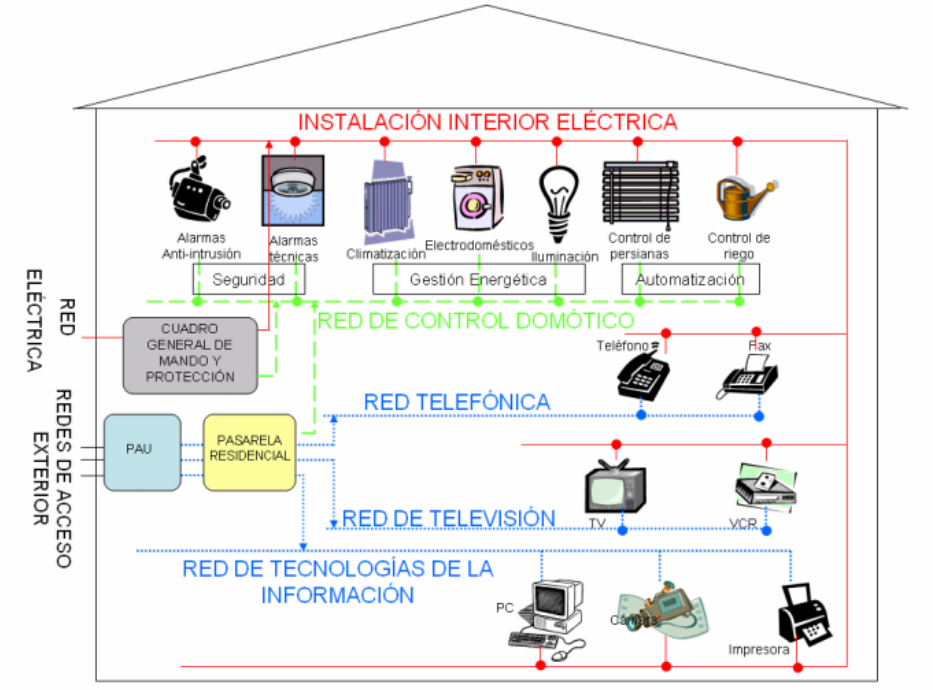


Ilustración 1. Redes de una instalación.

La instalación interior eléctrica (línea roja) y la red de control domótico (línea verde) están reguladas por el REBT. En particular, la red de control del sistema domótico está regulada por la ITC-BT-51 del REBT en lo referente a seguridad eléctrica y compatibilidad electromagnética.

Z-Wave

Z-Wave es un estándar internacional utilizado principalmente en domótica para el control mediante RF de los diferentes dispositivos que componen la red de sensores en un sistema domótico. Puede ser controlado a través de internet con una puerta de enlace Z-wave o servidor central.

Este sistema fue originalmente desarrollado por una joven empresa danesa llamada Zen-Sys y posteriormente adquirido por Sigma Designs en el año 2008. Existen en el mercado más de 1.500 productos fabricados funcionando con este protocolo, llegando a un número de ventas desde el 2005 de más de 35 millones de dispositivos.

Las características técnicas más importantes del protocolo Z-Wave son:

- Trabaja en el rango de frecuencias ISM alrededor de los 868,42 MHz (Europa).
- La codificación utilizada es Manchester.
- Utiliza una estructura de red tipo mesh.
- La velocidad de transmisión de datos (date-rate) es seleccionable entre de 9,6 kbit/s o 40 kbit/s, con baja latencia y alta fiabilidad.
- La MCU está basada en el chip ZW0201.
- Control mediante SPI.
- Funciona como transceptor.
- La transmisión se realiza mediante GFSK (*Gaussian Frequency Shift Keying* – Modulación por desplazamiento de frecuencia gausiana).

El sistema Arduino, *open source* y cultura *maker*

Arduino es un sistema de *hardware* libre o abierto que permite la implementación de funciones electrónicas variadas a través de sus diferentes módulos y software de programación o IDE *(Integrated Development Environment* – Entorno de Desarrollo Integrado). El lenguaje de programación de Arduino está basado en *Procesing* (muy similar a C++) y a través del IDE se realiza la carga del código en el microcontrolador Atmel que incorpora el sistema.

La idea de Arduino nació en el *Interaction Design Institue de* IVREA (Italia) cuando el entonces estudiante Massimo Banzi pensó en realizar un sistema mediante el que

los estudiantes pudieran desarrollar de forma económica y sencilla sus proyectos de diseño y programación de sistemas embebidos. Inicialmente estaba basado en una simple placa de circuitos eléctricos, donde estaban conectados un micro controlador simple junto con resistencias de voltaje, además de que únicamente podían conectarse sensores simples como leds u otras resistencias, y es más, aún no contaba con el soporte de algún lenguaje de programación para manipularla.

Años más tarde, se integró al equipo de Arduino Hernando Barragán, un estudiante de la Universidad de Colombia que se encontraba haciendo su tesis, y tras enterarse de este proyecto, contribuyó al desarrollo de un entorno para la programación del procesador de esta placa: *Wiring*, en colaboración con David Mellis, otro integrante del mismo instituto que Banzi, quien más adelante, mejoraría la interfaz de software.

Tiempo después, se integro el estudiante español David Cuartielles, experto en circuitos y computadoras, quien ayudó Banzi a mejorar la interfaz de hardware de esta placa, agregando los microcontroladores necesarios para brindar soporte y memoria al lenguaje de programación para manipular esta plataforma. Más tarde, Tom Igoe, un estudiante de Estados Unidos que se encontraba haciendo su tesis, escuchó que se estaba trabajando en una plataforma de *open-source* basada en una placa de micro controladores pre-ensamblada. Después se interesó en el proyecto y fue a visitar las instalaciones del Instituto IVRAE para averiguar en que estaban trabajando. Tras regresar a su país natal, recibió un e-mail donde el mismo Massimo Banzi invitó a Igoe a participar con su equipo para ayudar a mejorar Arduino. Aceptó la invitación y ayudó a mejorar la placa haciéndola más potente, agregando puertos USB para poder conectarla a un ordenador.

Gracias a la filosofía *open source* que lo sustenta, tanto en el modo en que se ensambla la placa con sus módulos o *shields*, como el funcionamiento de su IDE, como los códigos fuentes Arduino y su lenguaje de programación son de acceso público. Esto quiere decir que cualquiera de nosotros que quiera usarlo y/o mejorarlo pueda hacerlo. Al ser *open-hardware*, tanto su diseño como su distribución es libre, es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia.

La comunidad *maker* formada a su alrededor y la generación de habilidades compartida hace que la comunidad Arduino se desarrolle y enriquezca a partir del trabajo con la placa, la experimentación, la producción de conocimiento en torno a ella,

etc. y estas habilidades son compartidas dentro de la comunidad, pudiéndose accedida por cualquier persona.

La comunidad Arduino trabaja con multitud de bibliotecas de programación y al tratarse de un entorno de código abierto cualquier persona puede implementar una nueva biblioteca para compartir y enriquecer las capacidades colectivas. Podemos encontrar una gran variedad de estas bibliotecas en la página oficial de Arduino y en otras webs particulares de personas que comparten sus avances con el sistema.

La mayoría de los prototipos o placas Arduino trabajan con microcontroladores AVR *(Advanced Virtual RISC)* de la casa estadounidense Atmel. Estos controladores son una familia RISC (*Reduced Instruction Set Computing*) del fabricante americano y fueron diseñados en un principio para la ejecución eficiente de código C compilado, de ahí que el lenguaje Arudino sea tan parecido al código C. La familia de micro-controladores AVR es muy numerosa y podemos encontrar diferentes placas Arduino en función de su microprocesador Atmel. En la Tabla 1 se observan varios ejemplos de placas Arduino y sus microprocesadores entre otras características principales.

Modelo	Microcontrolador	Voltaje de entrada	Voltaje del sistema	Frecuencia de reloj	Entradas/salidas digitales	Entradas analógicas	PWM	UART	Memoria flash
Arduino Due	AT91SAM3X8E	5-12V	3,3V	84MHz	54	12	12	4	512Kb
Arduino Leonardo	ATmega32U4	7-12V	5V	16MHz	20	12	7	1	32Kb
Arduino Uno - R3	ATmega328	7-12V	5V	16MHz	14	6	6	1	32Kb
RedBoard	ATmega328	7-15V	5V	16MHz	14	6	6	1	32Kb
Arduino Pro 3.3V/8MHz	ATmega328	3,35 - 12V	3,3V	8MHz	14	6	6	1	32Kb
Arduino Pro 5V/16MHz	ATmega328	5 - 12V	5V	16MHz	14	6	6	1	32Kb
Arduino Mega 2560 R3	ATmega2560	7-12V	5V	16MHz	54	16	14	4	256Kb
Mega Pro 3.3V	ATmega2560	3,3-12V	3,3V	8MHz	54	16	14	4	256Kb
Mega Pro 5V	ATmega2560	5-12V	5V	16MHz	54	16	14	4	256Kb
Arduino Mini 05	ATmega328	7-9V	5V	16MHz	14	6	8	1	32Kb

Tabla 1. Varias características de algunas placas Arduino.

2 Esquema de la instalación.

Se propone la instalación del sistema domótico accesible desde portal WEB basado en la plataforma hardware y software libre Arduino sobre la vivienda mostrada en la Ilustración 2.

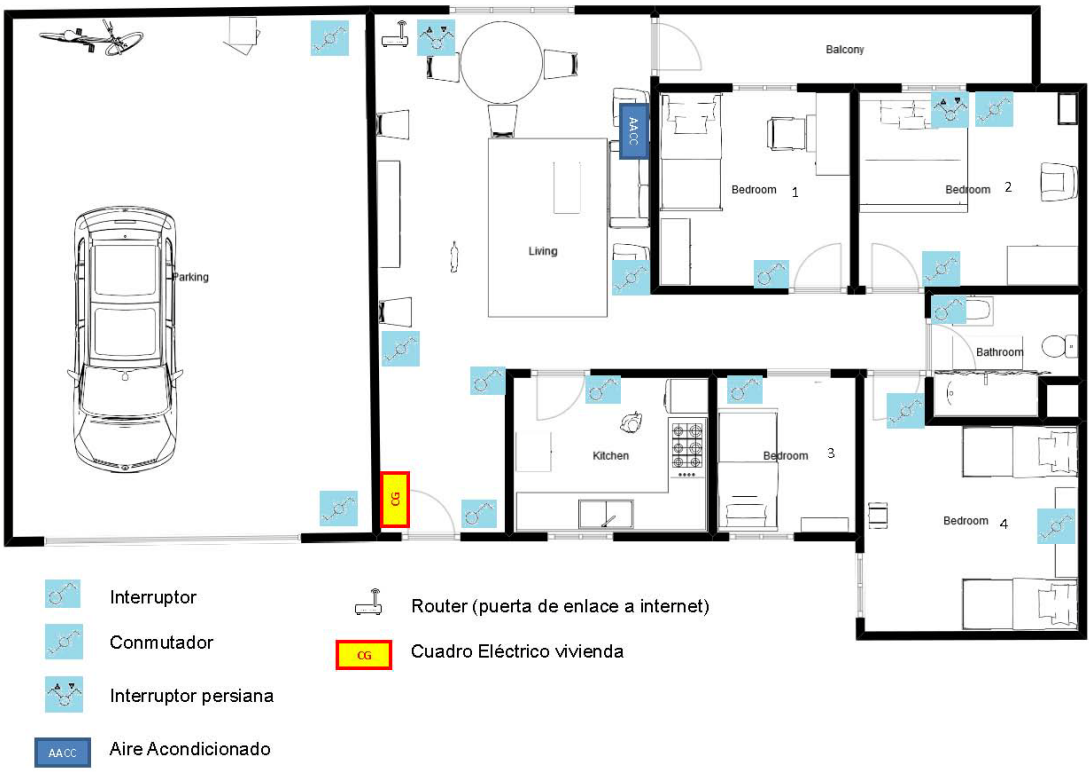


Ilustración 2. Situación actual de la vivienda.

La instalación del nodo central y los nodos remotos en función de la situación de cada uno de ellos se puede ver en la Ilustración 3. En el esquema podemos ver cada uno de los nodos, con la numeración que tendrán dentro del sistema, y que funcionalidad tendrán cada uno de ellos. El nodo central, o *master*, es el nodo “00” que se encargará de albergar el portal WEB y ser el enlace del sistema con internet. Podemos ver como tendremos diferentes funcionalidades como control de iluminación, temperatura, persianas, control AACC (aire acondicionado), estado de puertas y consumos.

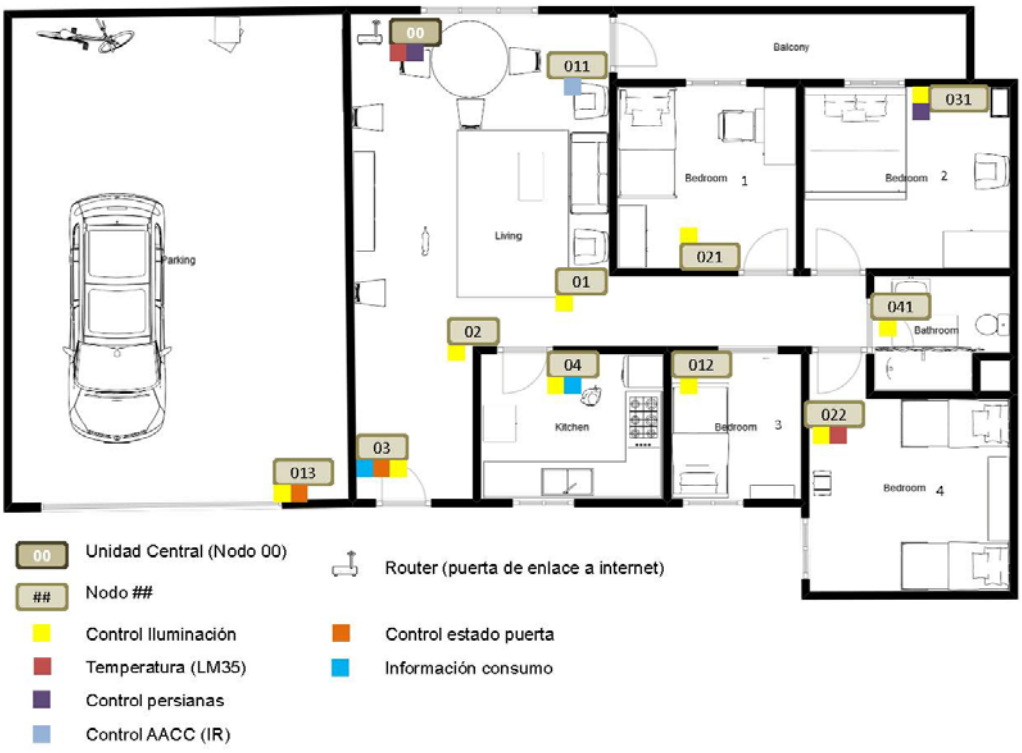


Ilustración 3. Esquema de la instalación en la vivienda.

3 Funciones del sistema.

A continuación se detallan las funciones que se pretende ofrecer con el sistema domótico en proyecto con el objetivo de fijar cada uno de los módulos que necesitaremos, indicando sus principales características y necesidades.

Las funciones que queremos ofrecer son:

- Acceso del sistema desde internet y desde la red wifi del hogar.
- Comunicación inalámbrica con los nodos de control.
- Portal web alojado en el sistema central.
- Control de la iluminación de todo el hogar.
- Información de la temperatura de diferentes estancias.
- Información del estado de las principales puertas del hogar (entrada principal y garaje).
- Información de consumo de los principales electrodomésticos.
- Interface para el futuro control del sistema de AACC (aire acondicionado).
- Interface para el futuro control de persianas.

En anexo nº 1 podemos ver el esquema general del funcionamiento del sistema con las funciones que ofrece.

3.1 Acceso del sistema desde internet y desde la red wifi del hogar.

El sistema ha de disponer de acceso a internet para poder ofrecer las funciones indicadas anteriormente y para ello utilizaremos el módulo Ethernet Shield en su versión 2.

Utilizaremos el módulo Ethernet Shield V2 para la comunicación mediante Ethernet del sistema Arduino y nuestra red local e internet. Lo utilizaremos para conectarlo al router y poder ofrecer el acceso WEB al sistema.

3.2 Comunicación inalámbrica con los nodos de control.

Para facilitar la instalación domótica en viviendas en las que no se ha previsto una preinstalación inicial (tubos y cajas de registro por donde pasar cables de comunicaciones y control), proponemos el uso de sistema de comunicación mediante RF en la banda libre de los 2.4 GHz.

El sistema central, formado por el módulo Arduino Mega 2560 hará las funciones de maestro gestionando las comunicaciones entre el resto de dispositivos, solicitando información y realizando peticiones cuando sea preciso. Los diferentes dispositivos que se pretende gobernar, actuando como esclavos, esperarán una comunicación del maestro y darán respuesta en caso necesario.

El módulo de RF propuesto para el presente proyecto es el módulo NRF24L01 que es capaz de desarrollar la capa de física mediante modulación GFSK (*Gaussian Frequency Shift Keying* – Modulación por desplazamiento de frecuencia gausiana), seleccionando entre 126 canales y con una velocidad de transmisión desde 250 Kbps hasta 2 Mbps.

3.3 Portal web alojado en el sistema central.

El servidor WEB se alojará en la memoria de programa del Arduino Mega (nodo central) facilitando una dirección IP y habilitando el puerto 80 (servicio HTTP) para poder ser contactado mediante comunicación TCP/IP a través del módulo Ethernet.

A través del protocolo de comunicación HTTP (*Hypertext Transfer Protocol*) enviaremos y recibiremos información, generando una página WEB mediante HTML (*HyperText Markup Language*) cargada en la memoria flash del Arduino y con las rutinas precisas que permitan el control del sistema desde el exterior.

Para poder comunicar con nuestro servidor WEB necesitaremos habilitar la función *port-forwarding* del router de la vivienda y conocer la dirección pública que el ISP (*Internet Service Provider*) nos asigna.

El acceso al sistema desde el exterior requiere conocer la IP pública del sistema. A través del acceso a la página http://ip.changeip.com obtendremos la IP pública. Sin embargo, conocer la IP pública en un momento dado no supone que dicha dirección se mantenga estática ya que los ISP (*Internet Service Provider*) habitualmente van modificando las IP's públicas de sus clientes.

A través del servicio DDNS (*Dinamic Domain Name Server)* obtendremos una URL direccionada a nuestra IP. El DDNS No-IP.com nos proporcionara de manera gratuita este servicio debiendo actualizar la IP al menos una vez al mes. De este modo podremos acceder al sistema a través de la URL facilitada sin preocuparnos de la IP pública y sus variaciones.

Existe otro método para conseguir solventar este problema. Contratar un servicio de IP estática con el proveedor de servicios de internet (ISP), que suele conllevar un suplemento económico por parte del ISP.

La implementación de la funcionalidad que permite actualizar la IP pública en el servicio DDNS se realizará mediante código de programa cargado en el sistema central conectado a internet (el Arduino Mega 2560). El programa detectará los cambios de IP pública accediendo a la web http://ip.changeip.com y transmitirá los datos necesarios mediante peticiones HTTP al servidor DDNS contratado para su actualización. De esta manera accediendo mediante el nombre de host, que siempre será estático, será posible enlazar con el sistema aunque la IP pública sufra cambios.

3.4 Control de la iluminación de todo el hogar (encendido y apagado).

El sistema nos permitirá conocer el estado de la iluminación del hogar por estancia, así como poder controlar el encendido y apagado de todas luces de la vivienda.

El uso de relés con dos contactos (NC y NA) servirá para integrar los actuales interruptores manuales de la vivienda con el sistema domótico.

Para conocer el estado de cada luz utilizaremos el sensor de corriente de efecto *hall* ACS712 que nos informará de la circulación de corriente a través de la línea de alimentación de las fuentes de luz.

3.5 Información de la temperatura de diferentes estancias.

Para tomar la medida de temperatura ambiental disponemos de diferentes sensores de temperatura que podemos utilizar con Arduino. Para el presente proyecto utilizaremos el LM35 como sensor.

3.6 Interface para futuro control del sistema de AACC.

Se dejará preparado el interface de comunicación entre el nodo central y el navegador que mostrará al usuario las principales funciones del sistema de AACC. También se preverá la comunicación entre el nodo central y un nodo remoto encargado del control de AACC. Quedará pendiente de desarrollar el código necesario para el accionamiento del aire acondicionado y la recogida de información de su estado.

En el apartado de líneas futuras se presenta un posible método para el control del sistema de aire acondicionado.

3.7 Interface para futuro control de persianas.

Del mismo modo que en la función de control de AACC se dejará preparado el interface de comunicación entre el nodo central y el navegado, y entre nodo central y nodo remoto para la implementación de un control de persianas.

En el apartado de líneas futuras se presenta un posible método para el control del de persianas.

3.8 Información del estado de puertas.

Podemos saber el estado de las puertas sin tener que modificar nada en ellas ni hacer ningún tipo de obra instalando un sistema de detección magnético, mediante interruptor, que informe a nuestro nodo Arduino del estado de la misma.

Ilustración 4. Interruptor magnético.

Mediante el uso de un interruptor magnético, en la modalidad N.A. (normalmente abierto) para ahorrar en consumo con la puerta cerrada (situación habtitual), podemos obtener el estado de puertas y/o ventanas.

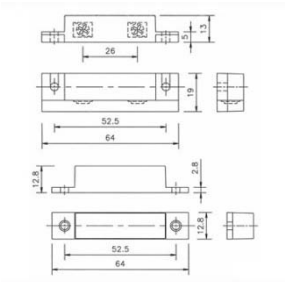


Ilustración 5. Dimensiones interruptor magnético NA.

3.9 Información de consumos.

Con el módulo ACS712 obtendremos información sobre la intensidad que circula por un circuito pudiendo calcular la potencia activa (Watios) y consumo (kWh).

Ilustración 6. Sensor de corriente ACS712.

4 Viabilidad económica del proyecto.

La elaboración del proyecto conlleva unos costes materiales y de recursos humanos con personal técnico para el desarrollo de los códigos, pruebas y puesta en marcha.

4.1 Presupuesto.

A continuación se presente el presupuesto detallado por nodo del proyecto con indicación del coste material para la instalación proyectada así como el coste de recursos humanos y del primer prototipo considerándolo como coste a amortizar.

		Cant	Precio ud.	Importe tot.
	Instalación en vivenda estandar			
	Costes materiales			
	Nodo 00 (salón)			
ud.	Arduino MEGA 2560	1	8,10 €	8,10 €
ud.	Arduino Ethernet Shield V2	1	5,58 €	5,58 €
ud.	NRF24l01	1	1,64 €	1,64 €
ud.	Fuente alimentación 9V - 1 A	1	8,26 €	8,26 €
ud.	Fuente alimentación 3.3V - 500 mA	1	2,48 €	2,48 €
ud.	Sensor de temperatura LM35DZ	1	1,66 €	1,66 €
ud.	Módulo de relé 220 V - 10 A (relé en cascada)	1	1,48 €	1,48 €
ud.	Módulo de relé 220 V - 30 A (persianas)	1	3,31 €	3,31 €
ud.	Pequeño material instalación (cableados y accesorios)	1	10,00 €	10,00 €
			Total	**42,50 €**
	Nodo 01 (luces salón)			
ud.	Arduino NANO	1	3,88 €	3,88 €
ud.	NRF24l01	1	1,64 €	1,64 €
ud.	Fuente alimentación 9V - 1 A	1	8,26 €	8,26 €
ud.	Fuente alimentación 3.3V - 500 mA	1	2,48 €	2,48 €
ud.	Módulo de relé 220 V - 10 A (luces)	1	1,48 €	1,48 €
ud.	Sensor de corriente ACS712 (detección estado luz)	1	3,06 €	3,06 €
ud.	Pequeño material instalación (cableados y accesorios)	1	10,00 €	10,00 €
ud.	Cambio conmutador por cruzamiento (llave de luz)	1	10,00 €	10,00 €
			Total	**40,79 €**
	Nodo 02 (luces pasillo)			
ud.	Arduino NANO	1	3,88 €	3,88 €
ud.	NRF24l01	1	1,64 €	1,64 €
ud.	Fuente alimentación 9V - 1 A	1	8,26 €	8,26 €
ud.	Fuente alimentación 3.3V - 500 mA	1	2,48 €	2,48 €
ud.	Módulo de relé 220 V - 10 A (luces)	1	1,48 €	1,48 €
ud.	Sensor de corriente ACS712 (detección estado luz)	1	3,06 €	3,06 €
ud.	Pequeño material instalación (cableados y accesorios)	1	10,00 €	10,00 €
ud.	Cambio interruptor por conmutador (llave de luz)	1	8,00 €	8,00 €
			Total	**30,79 €**
	Nodo 03 (entrada)			
ud.	Arduino NANO	1	3,88 €	3,88 €
ud.	NRF24l01	1	1,64 €	1,64 €
ud.	Fuente alimentación 9V - 1 A	1	8,26 €	8,26 €
ud.	Fuente alimentación 3.3V - 500 mA	1	2,48 €	2,48 €
ud.	Módulo de relé 220 V - 10 A (luces)	1	1,48 €	1,48 €
ud.	Sensor de corriente ACS712 (medida consumo general)	1	3,06 €	3,06 €
ud.	Sensor de corriente ACS712 (detección estado luz)	1	3,06 €	3,06 €
ud.	Contactos magnéticos	1	2,22 €	2,22 €
ud.	Pequeño material instalación (cableados y accesorios)	1	10,00 €	10,00 €
ud.	Cambio interruptor por conmutador (llave de luz)	1	8,00 €	8,00 €
			Total	**36,07 €**

		Cant	Precio ud.	Importe tot.
	Nodo 04 (cocina)			
ud.	Arduino NANO	1	3,88 €	3,88 €
ud.	NRF24l01	1	1,64 €	1,64 €
ud.	Fuente alimentación 9V - 1 A	1	8,26 €	8,26 €
ud.	Fuente alimentación 3.3V - 500 mA	1	2,48 €	2,48 €
ud.	Módulo de relé 220 V - 10 A (luces)	1	1,48 €	1,48 €
ud.	Sensor de corriente ACS712 (medida consumo general)	1	3,06 €	3,06 €
ud.	Sensor de corriente ACS712 (detección estado luz)	1	3,06 €	3,06 €
ud.	Pequeño material instalación (cableados y accesorios)	1	10,00 €	10,00 €
ud.	Cambio interruptor por conmutador (llave de luz)	1	8,00 €	8,00 €
			Total	**33,85 €**
	Nodo 011 (salón control AACC)			
ud.	Arduino NANO	1	3,88 €	3,88 €
ud.	NRF24l01	1	1,64 €	1,64 €
ud.	Fuente alimentación 9V - 1 A	1	8,26 €	8,26 €
ud.	Fuente alimentación 3.3V - 500 mA	1	2,48 €	2,48 €
ud.	Diodo LED IR	1	0,58 €	0,58 €
ud.	Pequeño material instalación (cableados y accesorios)	1	10,00 €	10,00 €
			Total	**26,83 €**
	Nodo 021 (habitación 1)			
ud.	Arduino NANO	1	3,88 €	3,88 €
ud.	NRF24l01	1	1,64 €	1,64 €
ud.	Fuente alimentación 9V - 1 A	1	8,26 €	8,26 €
ud.	Fuente alimentación 3.3V - 500 mA	1	2,48 €	2,48 €
ud.	Módulo de relé 220 V - 10 A (luces)	1	1,48 €	1,48 €
ud.	Sensor de corriente ACS712 (detección estado luz)	1	3,06 €	3,06 €
ud.	Pequeño material instalación (cableados y accesorios)	1	10,00 €	10,00 €
ud.	Cambio interruptor por conmutador (llave de luz)	1	8,00 €	8,00 €
			Total	**30,79 €**
	Nodo 031 (habitación 2)			
ud.	Arduino NANO	1	3,88 €	3,88 €
ud.	NRF24l01	1	1,64 €	1,64 €
ud.	Fuente alimentación 9V - 1 A	1	8,26 €	8,26 €
ud.	Fuente alimentación 3.3V - 500 mA	1	2,48 €	2,48 €
ud.	Módulo de relé 220 V - 10 A (luces)	1	1,48 €	1,48 €
ud.	Sensor de corriente ACS712 (detección estado luz)	1	3,06 €	3,06 €
ud.	Módulo de relé 220 V - 10 A (relé en cascada)	1	1,48 €	1,48 €
ud.	Módulo de relé 220 V - 30 A (persianas)	1	3,31 €	3,31 €
ud.	Pequeño material instalación (cableados y accesorios)	1	10,00 €	10,00 €
ud.	Cambio conmutador por cruzamiento (llave de luz)	1	10,00 €	10,00 €
			Total	**35,58 €**

		Cant	Precio ud.	Importe tot.
	Nodo 012 (habitación 3)			
ud.	Arduino NANO	1	3,88 €	3,88 €
ud.	NRF24l01	1	1,64 €	1,64 €
ud.	Fuente alimentación 9V - 1 A	1	8,26 €	8,26 €
ud.	Fuente alimentación 3.3V - 500 mA	1	2,48 €	2,48 €
ud.	Módulo de relé 220 V - 10 A (luces)	1	1,48 €	1,48 €
ud.	Sensor de corriente ACS712 (detección estado luz)	1	3,06 €	3,06 €
ud.	Pequeño material instalación (cableados y accesorios)	1	10,00 €	10,00 €
ud.	Cambio interruptor por conmutador (llave de luz)	1	8,00 €	8,00 €
			Total	**30,79 €**
	Nodo 022 (habitación 4)			
ud.	Arduino NANO	1	3,88 €	3,88 €
ud.	NRF24l01	1	1,64 €	1,64 €
ud.	Fuente alimentación 9V - 1 A	1	8,26 €	8,26 €
ud.	Fuente alimentación 3.3V - 500 mA	1	2,48 €	2,48 €
ud.	Sensor de temperatura LM35DZ	1	1,66 €	1,66 €
ud.	Módulo de relé 220 V - 10 A (luces)	1	1,48 €	1,48 €
ud.	Sensor de corriente ACS712 (detección estado luz)	1	3,06 €	3,06 €
ud.	Pequeño material instalación (cableados y accesorios)	1	10,00 €	10,00 €
ud.	Cambio conmutador por cruzamiento (llave de luz)	1	10,00 €	10,00 €
			Total	**32,45 €**
	Nodo 041 (cuarto de baño)			
ud.	Arduino NANO	1	3,88 €	3,88 €
ud.	NRF24l01	1	1,64 €	1,64 €
ud.	Fuente alimentación 9V - 1 A	1	8,26 €	8,26 €
ud.	Fuente alimentación 3.3V - 500 mA	1	2,48 €	2,48 €
ud.	Módulo de relé 220 V - 10 A (luces)	1	1,48 €	1,48 €
ud.	Sensor de corriente ACS712 (detección estado luz)	1	3,06 €	3,06 €
ud.	Pequeño material instalación (cableados y accesorios)	1	10,00 €	10,00 €
ud.	Cambio interruptor por conmutador (llave de luz)	1	8,00 €	8,00 €
			Total	**30,79 €**
	Nodo 013 (garaje)			
ud.	Arduino NANO	1	3,88 €	3,88 €
ud.	NRF24l01	1	1,64 €	1,64 €
ud.	Fuente alimentación 9V - 1 A	1	8,26 €	8,26 €
ud.	Fuente alimentación 3.3V - 500 mA	1	2,48 €	2,48 €
ud.	Módulo de relé 220 V - 10 A (luces)	1	1,48 €	1,48 €
ud.	Sensor de corriente ACS712 (detección estado luz)	1	3,06 €	3,06 €
ud.	Contactos magnéticos	1	2,22 €	2,22 €
ud.	Pequeño material instalación (cableados y accesorios)	1	10,00 €	10,00 €
ud.	Cambio conmutador por cruzamiento (llave de luz)	1	10,00 €	10,00 €
			Total	**33,02 €**
		Total materiales (IVA incluido)		**404,28 €**

		Cant	Precio ud.	Importe tot.
	Costes mano de obra y recursos humanos			
	Estudio de la situación y diseño del sistema			
Hora	Ing. Técnico en telecomunicaciones	16	30,00 €	480,00 €
	Incluye visita a la vivienda, recopilación de información de las caracterísitcas de la misma, tipo de router, ISP, elaboración de croquis, confección plano general en oficina y desarrollo del sistema basado en el prototipo incial			
			Total	**480,00 €**
	Instalación en la vivienda			
Hora	Ing. Técnico en telecomunicaciones	8	30,00 €	240,00 €
Hora	Técnico electricista	8	20,00 €	160,00 €
Hora	Técnico en electrónica de comunicaciones	32	20,00 €	640,00 €
			Total	**1.040,00 €**
	Finalización			
Hora	Técnico en electrónica de comunicaciones (pruebas)	8	20,00 €	160,00 €
			Total	**160,00 €**
			Total mano de obra y recursos humanos	**1.680,00 €**
			Costes de amortización del prototipo inicial	**400,00 €**
			Total instalación (materiales, mano de obra y amortización del prototipo)	**2.484,28 €**
			G.G. + B.I. (15%)	372,64 €
			P.V.P. IVA no inlcuido	2.856,92 €
			P.V.P. IVA (21%) inlcuido	**3.456,88 €**

El importe del precio de venta del producto instalado asciende a la cantidad de TRES MIL CUATROCIENTOS CINCUENTA Y SEIS EUROS CON OCHENTA Y OCHO CÉNTIMOS IVA incluido

En el importe de total de costes de la instalación se han tenido en cuenta los costes materiales, los costes de ingeniería e instalación, y una previsión de amortización del primer prototipo de 400 €.

El coste de inversión inicial se detalla a continuación. Podemos ver una inversión inicial de 160 horas (equivalente a 20 días de 8 horas) en la labor de investigación y desarrollo de los códigos a implementar en el sistema.

		Cant	Precio ud.	Importe tot.
	Amortización del prototipo inicial			
	Ingeniería e implantación primer prototipo (Coste único del primero prototipo)			
	Desarrollo prototipo y pruebas			
Hora	Ing. Técnico en telecomunicaciones	160	30,00 €	**4.800,00 €**
	Incluye desarrollo del código de carga en los microcontroladores: - Comunicación ethernet - Comunicación RF - Autenticación cifrada - Páginas WEB - Lógica de menus/control en nodo central y remotos - Control de sensores - Elaboración de manual básico Código totalmente preparado y probado para su funcionamiento final			
	Costes materiales para el desarrollo del protipo			
ud.	Arduino MEGA 2560	1	9,80 €	9,80 €
ud.	Arduino Ethernet Shield V2	1	6,75 €	6,75 €
ud.	NRF24l01	5	1,99 €	9,95 €
ud.	Fuente alimentación 9V - 1 A	5	9,99 €	49,95 €
ud.	Fuente alimentación 3.3V - 500 mA	5	3,00 €	15,00 €
ud.	Sensor de temperatura LM35DZ	2	2,01 €	4,02 €
ud.	Módulo de relé 220 V - 10 A (relé en cascada)	1	1,79 €	1,79 €
ud.	Módulo de relé 220 V - 30 A (persianas)	1	4,00 €	4,00 €
ud.	Arduino NANO	4	4,69 €	18,76 €
ud.	Módulo de relé 220 V - 10 A (luces)	4	1,79 €	7,16 €
ud.	Sensor de corriente ACS712 (detección estado luz)	4	3,70 €	14,80 €
ud.	Sensor de corriente ACS712 (medida consumo general)	1	3,70 €	3,70 €
ud.	Cambio interruptor por conmutador (llave de luz)	2	8,00 €	16,00 €
ud.	Cambio conmutador por cruzamiento (llave de luz)	2	10,00 €	20,00 €
ud.	Contactos magnéticos	1	2,69 €	2,69 €
ud.	Diodo LED IR	1	0,70 €	0,70 €
ud.	Pequeño material instalación (cableados y accesorios)	1	10,00 €	10,00 €
	Total materiales (IVA incluido)			**195,07 €**
	Total Ingeniería e implantación primer prototipo			**4.995,07 €**

Retorno de la inversión incial con la instalación de	12 instalaciones
Amortización por instalación	400,00 €

A partir de la 12 instalación podemos contar con 400 € menos de coste

Como podemos observar el coste final del producto no alcanza los 2.500 € (2.484,8 €) habiendo tenido en cuenta el coste de amortización del primer prototipo. Si consideramos no hacer frente al coste inicial del prototipo, sin contar la amortización de 400 €, podemos hablar de un coste 2.084,28 €, de los que 404,28 € son materiales

4.2 Otros sistemas similares en el mercado.

Existe un referente en el mercado de domótica inalámbrica y es el conocido Z-Wave que puede ser el sistema con más similitudes con el sistema proyectado en fase de desarrollo.

Z-Wave es el estándar internacional para la interconexión inalámbrica de los sistemas domóticos del hogar. Con este sistema se puede gestionar la iluminación, la electricidad, las persianas, las alarmas, etc.

Si accedemos a la página web http://zwave.es/ podemos ver una serie de productos con precios de compra sobre los que podemos realizar un estudio económico que represente un sistema similar al presentado en el presente proyecto.

A continuación se presenta un presupuesto estimado de los componentes obtenidos de la página Z-wave que componen un sistema de similares funcionalidades.

Z-wave http://zwave.es/

Código	Descripción	Función que implementa en comparación con el sistema protectado	uds.	Precio unitario	Total
ES-71-VEC_VERAEDGE-EU	Vera Edge	Nodo central - portal WEB	1	115,66 €	115,66 €
ES-22-ZIP_PAN03	Micromódulo conmutador 1x2.5 kW con medida de energía Zipato Z-wave +	Control iluminación	10	48,76 €	487,60 €
ES-61-PHI_PAN08	Módulo oculto de persianas Philio	Control persiana	2	47,93 €	95,86 €
ES-11-FIB_FGBS_001	Sensor Binario universal de Fibaro	Sensor de temperatura	2	33,05 €	66,10 €
ES-43-DAL_DS18B20	Sensor de temperatura DS18B20		2	3,00 €	6,00 €
ES-13-EVR_SM103White	Detector de apertura EverSpring Blanco	Detector de puerta abierta	2	34,50 €	69,00 €
ES-51-EVR_AN1582	Enchufe controlado (on/off) con medidas de consumos Everspring	Medidor de consumo	2	43,79 €	87,58 €

P.V.P. materiales (IVA no incluido)	**927,80 €**
IVA (21%)	**194,84 €**
P.V.P. materiales (IVA no incluido)	**1.122,64 €**

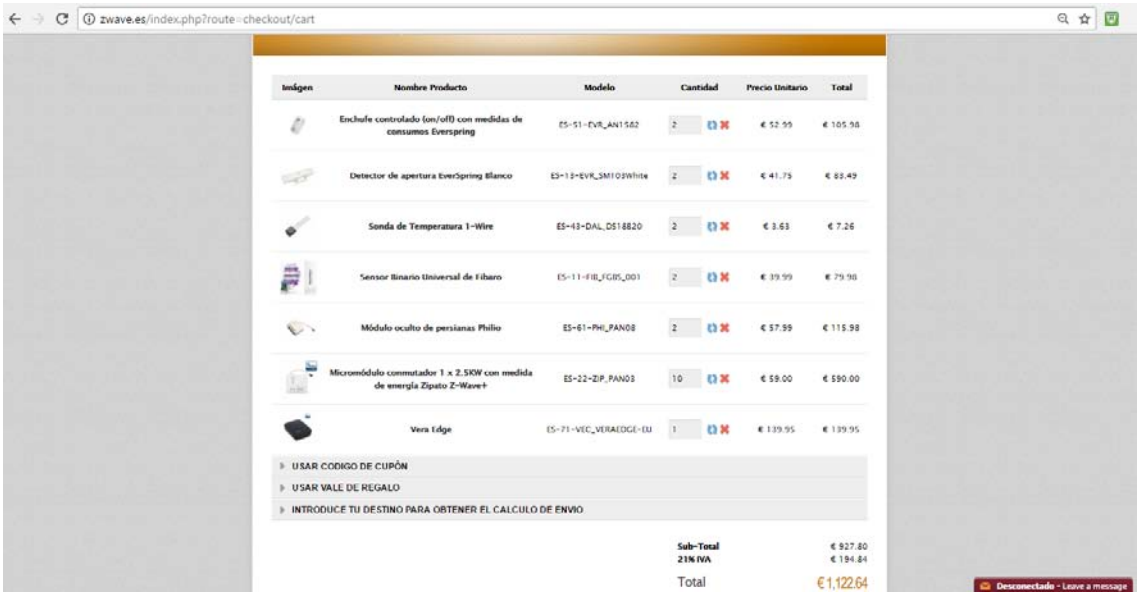


Ilustración 7. Presupuesto de materiales de una instalación similar.

Comparando los importes de los materiales del proyecto y los materiales de los elementos seleccionados en zwave.es observamos una diferencia significativa, del orden del doble.

Materiales zwave.es	927,80 €
Materiales proyecto más 15% (GG y BI)	464,92 €
Diferencia (sin IVA)	462,88 €

Sin embargo, debemos tener en cuenta que el producto final obtenido con el presente proyecto no contempla el diseño de producto final con cajas diseñadas para albergar la electrónica de cada nodo, las placas PCB que contengan todos los elementos de cada nodo en una única placa, y las funciones complementarias que ofrecen los productos Z-Wave.

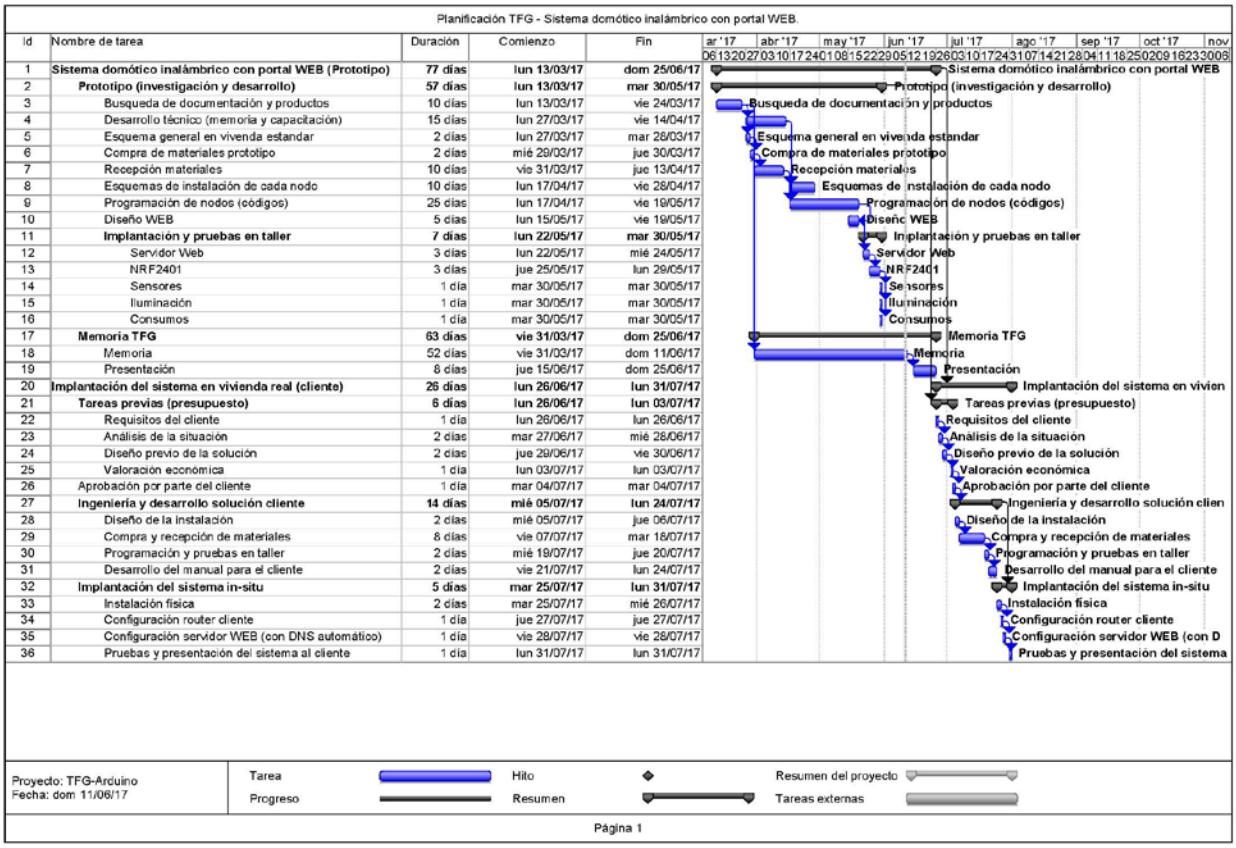
Podemos encontrar en el mercado una gran variedad de cajas de plástico para contener los circuitos en su interior, de varias dimensiones y colores. El precio medio de las cajas puede rondar los 10 €. Teniendo en cuenta que nuestro sistema precisaría el uso de 12 cajas, una por cada nodo, tendríamos un coste aproximados por las cajas de 120 €.

Contemplando un posible coste de diseño final, encapsulados, mejora de funcionalidades, etc. y contando con una reducción en los costes materiales por volúmenes y gestiones de compra, vemos como es posible llegar a poner en el mercado el producto final.

En cualquier caso el objetivo principal del TFG no es comercial, sino académico y en busca un beneficio común de una comunidad que comparte contenidos, inteligencia, proyectos, por lo que no se busca un producto comercial sino un producto económico y funcional.

5 Planificación.

A continuación se muestra la planificación seguida en el desarrollo del proyecto y confección de la presente memoria. En la planificación podemos ver un apartado destinado a la posible implantación del sistema en la vivienda reflejada en proyecto.



Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Sistema domótico inalámbrico con portal WEB (Prototipo)**	**77 días**	**lun 13/03/17**	**dom 25/06/17**
2	**Prototipo (investigación y desarrollo)**	**57 días**	**lun 13/03/17**	**mar 30/05/17**
3	Busqueda de documentación y productos	10 días	lun 13/03/17	vie 24/03/17
4	Desarrollo técnico (memoria y capacitación)	15 días	lun 27/03/17	vie 14/04/17
5	Esquema general en vivienda estandar	2 días	lun 27/03/17	mar 28/03/17
6	Compra de materiales prototipo	2 días	mié 29/03/17	jue 30/03/17
7	Recepción materiales	10 días	vie 31/03/17	jue 13/04/17
8	Esquemas de instalación de cada nodo	10 días	lun 17/04/17	vie 28/04/17
9	Programación de nodos (códigos)	25 días	lun 17/04/17	vie 19/05/17
10	Diseño WEB	5 días	lun 15/05/17	vie 19/05/17
11	**Implantación y pruebas en taller**	**7 días**	**lun 22/05/17**	**mar 30/05/17**
12	Servidor Web	3 días	lun 22/05/17	mié 24/05/17
13	NRF2401	3 días	jue 25/05/17	lun 29/05/17
14	Sensores	1 día	mar 30/05/17	mar 30/05/17
15	Iluminación	1 día	mar 30/05/17	mar 30/05/17
16	Consumos	1 día	mar 30/05/17	mar 30/05/17
17	**Memoria TFG**	**63 días**	**vie 31/03/17**	**dom 25/06/17**
18	Memoria	52 días	vie 31/03/17	dom 11/06/17
19	Presentación	8 días	jue 15/06/17	dom 25/06/17
20	**Implantación del sistema en vivienda real (cliente)**	**26 días**	**lun 26/06/17**	**lun 31/07/17**
21	**Tareas previas (presupuesto)**	**6 días**	**lun 26/06/17**	**lun 03/07/17**
22	Requisitos del cliente	1 día	lun 26/06/17	lun 26/06/17
23	Análisis de la situación	2 días	mar 27/06/17	mié 28/06/17
24	Diseño previo de la solución	2 días	jue 29/06/17	vie 30/06/17
25	Valoración económica	1 día	lun 03/07/17	lun 03/07/17
26	Aprobación por parte del cliente	1 día	mar 04/07/17	mar 04/07/17
27	**Ingeniería y desarrollo solución cliente**	**14 días**	**mié 05/07/17**	**lun 24/07/17**
28	Diseño de la instalación	2 días	mié 05/07/17	jue 06/07/17
29	Compra y recepción de materiales	8 días	vie 07/07/17	mar 18/07/17
30	Programación y pruebas en taller	2 días	mié 19/07/17	jue 20/07/17
31	Desarrollo del manual para el cliente	2 días	vie 21/07/17	lun 24/07/17
32	**Implantación del sistema in-situ**	**5 días**	**mar 25/07/17**	**lun 31/07/17**
33	Instalación física	2 días	mar 25/07/17	mié 26/07/17
34	Configuración router cliente	1 día	jue 27/07/17	jue 27/07/17
35	Configuración servidor WEB (con DNS automático)	1 día	vie 28/07/17	vie 28/07/17
36	Pruebas y presentación del sistema al cliente	1 día	lun 31/07/17	lun 31/07/17

Ilustración 8. Planificación.

6 Arduino Mega 2560.

La unidad de procesamiento del sistema central estará formada por el MCU ATMega2560 sobre una placa Arduino Mega2560 R3. Este micro dispone de 54 entradas-salidas digitales, de las cuales 14 pueden utilizarse como salidas PWM (*Pulse width modulation* – Modulación por ancho de pulso), 16 entradas analógicas, 4 puertos UARTs (*Universal Asynchronous Receiver-Transmitter* – Receptor-Transmisor Asíncrono Universal) para comunicaciones serie, su frecuencia de reloj es de 16 MHz y dispone de 256 KB de memoria FLASH, 8 KB SRAM y 4KB EEPROM.

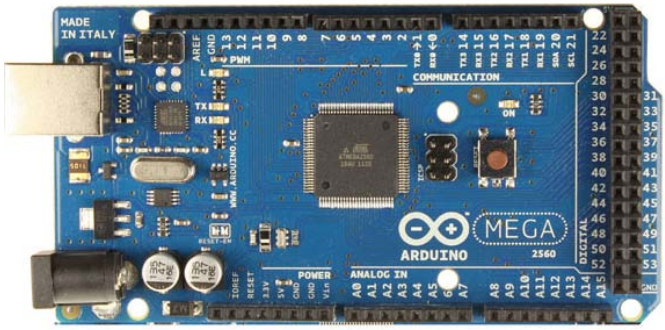


Ilustración 9. Placa Arduino Mega 2560 R3

Cada uno de los pines digitales de la placa Arduino puede ser utilizado como una entrada o salida, utilizando funciones pinMode(), digitalWrite(), y digitalRead() en el código de programación. Operan en 5 V. Cada pin puede proporcionar o recibir un máximo de 40 mA y tiene una resistencia de *pull-up* (desconectado por defecto) de 20 a 50 kOhm. Además, algunos pines tienen funciones especializadas (MEGA 2560):

- **Serial**: Pines 0, 1, 14, 15, 16, 17, 18 y 19. (RX) y (TX) de los 4 UART's. Se utilizan para recibir (RX) y transmitir datos en serie (TX) TTL.
- **Interrupciones externas**: Pines 2, 3, 21, 20, 19 y 18, correspondientes a las interrupciones externas 0, 1, 2, 3, 4 y 5 respectivamente. Estos pines pueden configurarse para activar una interrupción en un valor bajo, un flanco ascendente o descendente, o un cambio en el valor.
- **PWM**: Corresponden a los pines del 2 al 13, proporcionan una salida PWM de 8 bits con la función analogWrite().
- **SPI**: Corresponden a los pines 53 (SS), 51 (MOSI), 50 (MISO) y 52 (SCK) estos pines admiten la comunicación SPI. *Serial Peripheral Interface* (SPI) es un protocolo de datos en serie síncrono utilizado por los microcontroladores

para comunicarse con uno o más dispositivos periféricos rápidamente en distancias cortas. También se puede utilizar para la comunicación entre dos microcontroladores.

- **LED**: 13. Hay un LED incorporado conectado al pin digital 13.

7 Arduino NANO.

Para el control y comunicación de los diferentes dispositivos que se pretende gobernar se utilizarán placas Arduino Nano con modulo NRF24L01. Este prototipo Arduino dispone solamente de 32 KB de memoria FLASH frente a los 256 KB del modelo MEGA 2560, sin embargo será suficiente debido a que el código con el que se programarán los Arduino Nano será menos pesado al no tener que soportar los algoritmos de comunicación Ethernet, Servidor WEB, algoritmos de estado, servicios y comunicación. Podemos ver un prototipo de Arduino Nano en la **¡Error! No se encuentra el origen de la referencia.**.

Las principales características del Arudino Nano son:

- Micro ATmega 328.
- 14 pines de entrada/salida digital.
- 6 pines (de los 14 anteriores) que pueden ser utilizados con PWM para simular salidas analógicas.
- 8 entradas analógicas.
- 1 puerto UART.
- Frecuencia del reloj de 16 MHz.
- 32 KB de memoria FLASH.
- 2 KB SRAM.
- 1 KB EEPROM.

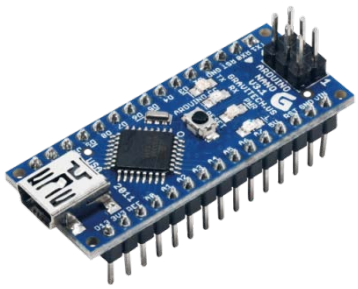


Ilustración 10. Arduino Nano V3.

8 Ethernet Shield.

Ethernet Shield V1

La versión 1.0 del shield actualmente está retirada y la documentación existente en la web no se actualiza.

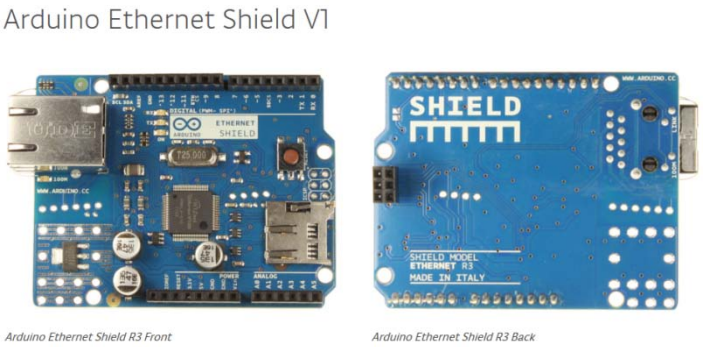


Ilustración 11. Arduino Shield v1.

A pesar de que es un producto que está descatalogado ya por la aparición de su sucesor, el módulo V2, todavía es posible encontrar módulos V1 en el mercado.

Las características principales del módulo V1, son:

- Trabaja con 5V DC.
- Opera con el chipset WIZNET W5100 con 16KB de buffer interno.
- Contiene un módulo de expansión de memoria dinámica micro SD.
- Soporte 10/100 Mb.
- Se conecta con Arduino a través del puerto SPI (*Serial Peripheral Interface*)
- Tiene un jack estándar RJ-45.
- Permite trabajar con PoE (con un módulo extra).
- Soporta protocolo de transporte TPC y UDP.
- Soporta 4 sockets de conexión simultáneos.

Sin embargo, este módulo no funciona correctamente con el Arduino Mega 2560 al no reiniciarse cuando se enciende la placa Arduino debiéndose reiniciar manualmente, lo que puede suponer un gran inconveniente dejando sin comunicación el sistema. La versión V2 ya incorpora un control de reinicio compatible con Arduino Mega que funciona de forma automática.

Ethernet Shield V2

Tal como se ha comentado en el apartado anterior en módulo V2 funciona correctamente con Arduino Mega reiniciando el sistema Ethernet a la vez que se reinicia el Arduino lo que evita problemas de conexión a la red.

Ilustración 12. Ethernet Shield V2.

Las características principales del módulo V2, son:

- Trabaja con 5V DC.
- Opera con el chipset WIZNET W5500 con 32KB de buffer interno.
- Contiene un módulo de expansión de memoria dinámica micro SD.
- Soporte 10/100 Mb.
- Se conecta con Arduino a través del puerto SPI (*Serial Peripheral Interface*)
- Tiene un jack estándar RJ-45.
- Permite trabajar con PoE (con un módulo extra).
- Soporta protocolo de transporte TPC y UDP.
- Soporta 8 sockets de conexión simultáneos.

Debido, principalmente, a la necesidad de utilizar un Arduino Mega 2560 como controlador principal que contenga el código de programa en su memoria flash de 256 KB, frente a la memoria flash de 32 KB de Arduino Uno, deberemos utilizar el módulo Ethernet Shield V2 y evitar problemas con los reinicios del sistema.

8.1 El módulo Ethernet Shield V2 simplificado.

El módulo Ethernet Shield V2 que utilizaremos será un módulo simplificado con las funcionalidades Ethernet pero sin la expansión de tarjeta SD. Los datos correspondientes a las páginas WEB (código HTML) que se utilizaran como interface entre el usuario y el sistema estarán almacenados en la memoria disponible en el microcontrolador Atmel ATMega2560.

Ilustración 13. Ethernet Shield V2 simplificado.

Las características principales son:

- Chipset: W5500
- Soporta alimentación a 3.3V y 5V.
- Protocolos soportados: TCP, UDP, ICMP, IPv4, ARP, IGMP, PPPoE.
- 10BaseT/100BaseTX Ethernet.
- Supports automatic response (full duplex / half duplex mode).
- Soporta simultáneamente 8 sockets independientes.
- 32Kbytes de memoria internar para buffers Tx/Rx.
- SPI interface (SPI MODE 0, 3).
- Tamaño: 55mm x 28mm.

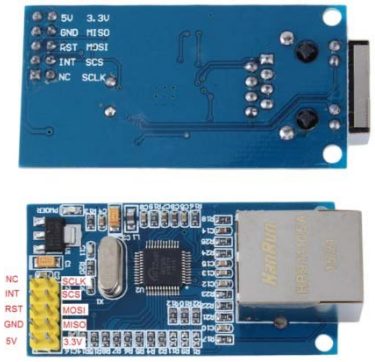


Ilustración 14. Pines SPI en Ethernet V2.

Este módulo trabaja con el microcontrolador WZ5500 de Wiznet cuyo diagrama de bloques podemos ver en la Ilustración 15. Este controlador provee al sistema de una pila de red IP capaz de soportar TCP y UPD, soporta hasta ocho conexiones de sockets simultáneas y permite trabajar con el estándar IEEE 802.3 10BASE-T y 802.3u 100BASE-TX.

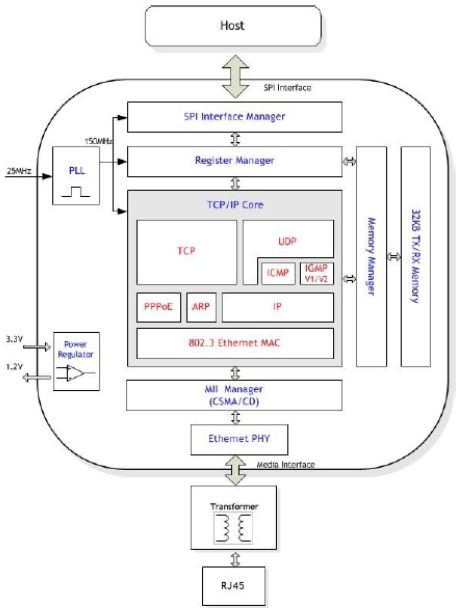


Ilustración 15. Diagrama de bloques operativo del chip WZ5500.

8.2 Librería Ethernet2.

Podemos encontrar la librería Ethernet2 en la web oficial de Arduino (Arduino.cc 2016a).

Con la utilización de la librería ethernet2.h controlaremos el funcionamiento del módulo Ethernet V2 y obtendremos datos para ser procesados por el MCU ATMega 2560.

La siguiente tabla muestra las principales clases y métodos que pueden ser utilizados con la librería *ethernet.h* y/o *ethernet2.h.*

Clase	**Método**	**Descripción**
Ethernet	begin()	Inicializa la librería Ethernet y la configuración de la red
	localip()	Obtiene la dirección IP del *shield*. Se utiliza en redes con DHCP
	maintain()	Para pedir renovación de IP al servidor DHCP (habitualmente al router)
IPAddress	IPAddress()	Para definir direcciones IP (formato IPv4: x, x, x, x en base 10)
Server	EthernetServer()	Permite instancia servidor nuevo que escuche en un puerto determinado
	begin()	Inicia la escucha de un servidor dado.
	avaible()	Devuelve un objeto Ethernet.client si llega una petición al servidor.
	write()	Escribe datos (*bytes*) a todos los clientes conectados al servidor.
	print()	Escribe datos (como secuencia de dígitos) a todos los clientes conectados.
	println()	Escribe datos seguidos de un fin de línea (como secuencia de dígitos) a todos los clientes conectados.
Client	EthernetClient()	Crea un nuevo objeto tipo cliente.
	If(EthernetClient)	Indica si un cliente está preparado.
	connected()	Indica si un cliente está conectado.

	connect()	Conectar a una dirección específica y puerto.
	write()	Escribe datos (en *bytes*) al servidor al que está conectado el cliente.
	print()	Escribe datos (como secuencia de dígitos) al servidor al que está conectado el cliente.
	println()	Escribe datos seguidos de un fin de línea (como secuencia de dígitos) al servidor al que está conectado el cliente.
	avaible()	Devuelve el número de *bytes* que han sido enviados al cliente por un servidor.
	read()	Lee el siguiente byte recibido, después de la última llamada de lectura.
	flush()	Espera hasta que todos los *bytes* en el *buffer* hayan sido enviados.
	stop()	Desconecta del servidor.

Tabla 2. Principales clases y métodos de la librería ethernet v2.

9 NRF24L01.

9.1 El módulo NRF24L01.

Mediante el uso del modulo NRF24L01 podremos dotar a cada MCU Arduino de capacidad de comunicación mediante RF.

El módulo NRF2401, basado en el chip de Nordic Semiconductor, proporcionará comunicación inalámbrica a cada MCU (maestro y esclavos. La comunicación se realizará mediante modulación GFSK a 2.4 GHz y permitirá seleccionar entre 126 canales diferentes entorno a los 2.4 GHz. La velocidad de transmisión puede ser seleccionada entre 250 Kbps, 1 Mbps o 2Mbps. Dado que los datos que se transmitirán serán órdenes o respuestas de estado con una carga de datos muy baja (del orden de 4 bytes) el sistema proyectado funcionará a 250 Kbps.

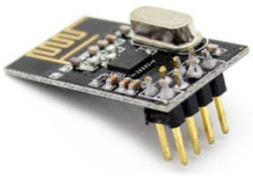


Ilustración 16. Placa NRF24L01.

El microchip NRF2401 funciona como transceptor de RF a 2.4 GHz. No puede emitir y recibir al mismo tiempo por lo que es necesario gestionar los tiempos de transmisión y recepción por parte del sistema central. Trabaja con corrección de errores (FEC) y protocolo de reenvío cuando es preciso. Funcionan a 3.3 V y necesita una cantidad de intensidad superior a la que puede proporcionar la placa Arudino para el arranque del módulo por lo que será preciso la alimentación mediante fuente

externa de 3.3 V, sin embargo sí puede trabajar con señales de 5V en su puerto SPI. La comunicación entre el NRF2401 y Arduino se realiza mediante SPI (*Serial Peripheral Interface*).

A continuación podemos ver el diagrama de bloques del chip nRF24L01 de Nordic Semiconductor.

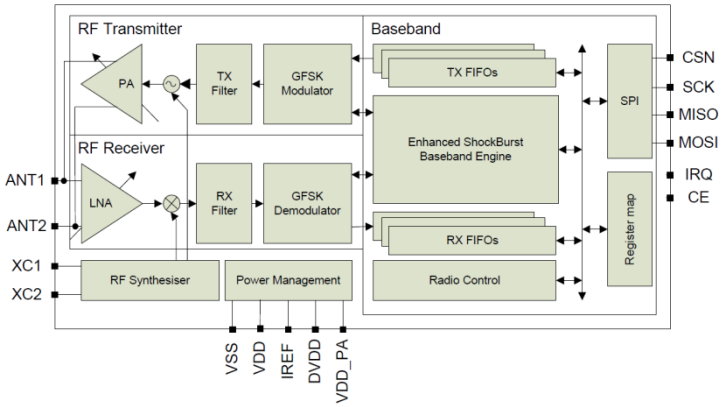


Ilustración 17. Diagrama de bloques NRF24L01.

Este chip viene integrado en una placa con los elementos necesarios para controlar la frecuencia de trabajo con un cristal de cuarzo de 16MHz, antena sobre pista integrada y componentes SMD para hacer funcionar el microchip, además de un puerto para las comunicaciones SPI.

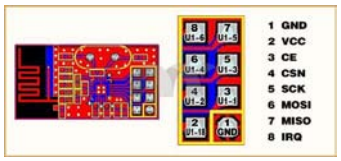


Ilustración 18. Pines SPI en módulo NRF24L01.

A continuación detallamos cada uno de los pines dispuestos en la placa:

- 1. GND.
- 2. VCC. Que ha de ser de 3.3 VDC, aunque puede oscilar entre +1.9 VDC y +3.6 VDC.
- 3. CE. *Chip Enable.* Permite seleccionar entre modo RX o TX.
- 4. CSN. Chip Select N para selección en el protocolo SPI.
- 5. SCK. Señal de reloj SPI.
- 6. MOSI. *Master Out, Slave In*. SPI.
- 7. MISO. *Master In, Slave Out.* SPI.

- 8 IRQ. Puerto de interrupción. Se puede utilizar para captar algún evento producido en el módulo de RF.

En documentación complementaria se adjuntan las especificaciones del chip nRF24L01 donde es posible consultar los modos de operación, registros, consumos, valores máximos de alimentación, etc.

9.2 Librerías RF24.

Existen varias librerías que desempeñan diferentes funciones para el manejo del módulo NRF24l01.

En este proyecto se utilizarán las librerías RF24.h y RF24Network.h que podemos encontrar en la web de GitHub (GitHub 2017) y que son facilitadas como documentación complementaria.

Algunas de las funciones que podremos utilizar para manejar el módulo NRF2401 (trabajando con la librería RF24.h) se muestran en la **¡Error! No se encuentra el origen de la referencia.**.

Clase	Método	Descripción
RF24	RF24(uint8_t _cepin, uint8_t _cspin)	Constructor de la clase RF24. Crea una nueva instancia del driver NRF2401 para la comunicación Arduino-NRF2401.
	begin()	Inicializa la comunicación entre Arduino y el módulo NRF2401
	startListening()	Comenzar la escucha en las *pipes* (canales) abiertos para lectura.
	stopListening()	Para la escucha.
	avaible()	Chequear cuando hay datos disponibles para leer.
	read(void* buf, uint8_t len)	Leer los datos disponibles. Se almacenan a partir de la dirección marcada por puntero.
	write (const void *buf, uint8_t len)	Para mandar los datos almacenados a partir de la dirección marcada por el puntero.
	openWritingPipe (const uint8_t *address)	Abrir la conexión en la dirección marcada (canal). Para escritura.
	openReadingPipe (uint8_t number, const uint8_t *address)	Abrir la conexión en la dirección marcada (canal). Para lectura.

Tabla 3. Principales clases y métodos de la librería RF24.

Con librería RF24Netowrk podemos implementar la capa de red del modelo OSI con los microcontroladores de radio RF24. La función de la capa de red es enviar datos desde un nodo de origen hasta un nodo destino encargándose de gestionar la

transmisión de paquetes a través de la red pasando por varios nodos hasta llegar al destino.

Una de sus principales funciones de la RF24Nework.h es el enrutamiento o encaminamiento que se encarga de buscar un camino entre todos los posibles en una red. Podemos ver varios ejemplos del encaminamiento de paquetes en la **¡Error! No se encuentra el origen de la referencia.**.

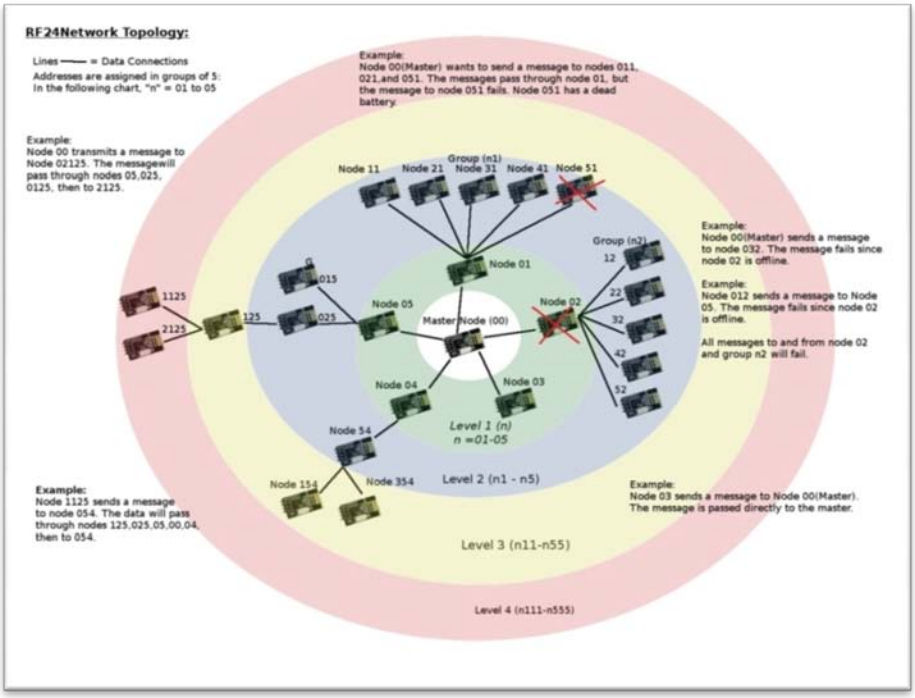


Ilustración 19. Ejemplos de transmisión en una red con RF24Network.

Cuando se realiza una transmisión desde un módulo RF a otro, el módulo receptor devuelve un paquete de reconocimiento (ACK) para indicar que ha sido recibido. SI el emisor no recibe el ACK, la radio automáticamente reenvía el paquete transcurrido un tiempo establecido para el reenvío. La radio usa técnicas de direccionamiento y numeración de paquetes para poder controlar los reenvíos y fallos. Los módulos NRF24L01 realizan automáticamente el reenvío y el acuse de recibo entre conexiones directas de manera transparente al usuario, pero cuando hay comunicación entre nodos que no están directamente conectados, sino a través de otro nodo (por ejemplo del nodo 00 al nodo 031), la capa de red se encarga de gestionar el reenvío y acuses de recibo (ACK's), y para ello se utilizan los tipos de mensajes de las cabeceras.

Las comunicaciones en la red creada a través de la librería RF24Ethernet se realizan mediante la asignación de *pipes* (túneles o circuitos virtuales) entre nodos mediante sus direcciones. Cada radio escucha hasta 6 direcciones en 6 *pipes* (una en

cada *pipe*), y cada radio tiene un nodo padre y 5 hijos (cada uno con su *pipe*), con lo que se puede formar en red tipo estructura. Los nodos se comunican directamente con sus padres e hijos, y la comunicación entre otros nodos se realiza mediante el enrutamiento que realiza la librería.

La tipología de red en modo estructura necesita seguir un direccionamiento adecuado, y de manera similar al direccionamiento IP de una red Ethernet, la librería RF24Ethernet necesita que sus nodos sean direccionados para manejar el enrutamiento. El direccionamiento de red para los módulos NRF24l01 sigue este modelo:

- Nodo master "00".
- Nodos hijos del master: "01", "02", "03", "04", "05"
- Nodos hijos del nodo "01": "011", "021", "031", "041", "051"

El enrutamiento de los paquetes es realizado por la capa de red de manera transparente para el usuario (implementado en la librería). El usuario simplemente ha de construir una cabecera que contenga la dirección de destino y la red transportará el paquete al destino correcto.

A continuación (Tabla 4) se muestran algunas de las funciones básicas de la clase RF24Network que utilizaremos en el proyecto.

Clase	**Método**	**Descripción**
RF24Network	RF24Network (RF24 &_radio)	Constructor de la clase RF24Network. Crea una nueva instancia de la red con el dispositivo RF24.
	begin (uint16_t _node_address)	Da comienzo al funcionamiento de la red como nodo indicado en "_node_address".
	uint8_t update (void)	Refresca los nodos que hay en la red. Es conveniente llamar a esta función cada cierto tiempo. Devuelve el tipo de mensaje de la última trama.
	bool available (void)	Nos indica si hay algún mensaje disponible para el nodo.
	uint16_t peek (RF24NetworkHeader &header)	Lee la siguiente cabecera disponible sin avanzar al siguiente mensaje entrante. Útil para hacer un cambio en el tipo de mensaje Si no hay ningún mensaje disponible, no cambia la cabecera pasada como parámetro "&header".
	uint16_t read (RF24NetworkHeader &header, void *message, uint16_t maxlen)	Lee un mensaje existente en la cabecera pasada como parámetro y lo almacena en el espacio de memoria apuntado por la variable "message". Maxlen para indicar la capacidad máxima que se puede copiar en memoria. Devuelve el número de bytes que han sido copiados en "message".
	bool write (RF24NetworkHeader &header, const void *message, uint16_t len)	Envía un mensaje con la cabera &header, mensaje en la variable "message" con una longitud len. Devuelve true cuando el mensaje ha sido recibido correctamente.

Tabla 4. Funciones básica de RF24Network.

La capa de red del modelo OSI

La capa de red es la encargada de gestionar el enrutamiento y el envío de paquetes entre los nodos o redes. La función principal de la capa de red es transferir datos desde el host que origina los datos hacia el host de destino, a través de varios nodos o redes separadas si fuera preciso.

Para realizar el transporte de datos de extremo a extremo la capa de red realiza cuatro procesos básicos:

- Direccionamiento: existe un direccionamiento de los dispositivos.
- Encapsulamiento: la información de la capa superior es encapsulada en paquetes de la capa de red con una dirección de destino.
- Enrutamiento: se selecciona el camino que debe seguir un paquete desde su origen hasta su destino.
- Desencapsulamiento: finalmente el paquete llega al destino. El receptor examina que efectivamente el paquete es para él, mediante la dirección del encabezado, y seguidamente extrae la información. Existen métodos de comprobación de la integridad del paquete como por ejemplo un CRC del paquete.

Para que un dispositivo de la capa de red sea capaz de reenviar el paquete que recibe hacia su destinatario, es necesario que este dispositivo tenga un “mapa de red” que le permita saber el camino que ha de seguir un paquete. Este proceso es el denominado enrutamiento.

Un datagrama es un paquete de datos que constituye el mínimo bloque de información en una red de conmutación por datagramas no orientada a conexión. La alternativa a esta conmutación de paquetes es el circuito virtual, orientado a conexión (la capa de red ofrece dos modos de funcionamiento, uno orientado a conexión y otro no orientado a conexión).

En la técnica de datagramas (modo no orientado a conexión) cada paquete se trata de forma independiente gracias a la cabecera que contiene información de origen, destino, fragmentación, etc… que permite su tratamiento en destino.

Un paquete de datos o datagrama es una unidad fundamental de transporte de información. Generalmente está compuesto por tres elementos:

- Cabecera (*header*): contiene la información necesaria para trasladar el paquete desde el emisor hasta el receptor.
- Datos (*payload*): contiene los datos.
- Cola (*trailer*): contiene un código de detección de errores.

Dependiendo de si se trata de una red de datagramas o de una red de circuitos virtuales (orientada a conexión), la cabecera del paquete contendrá la dirección de las estaciones de origen y destino o el identificador del circuito virtual.

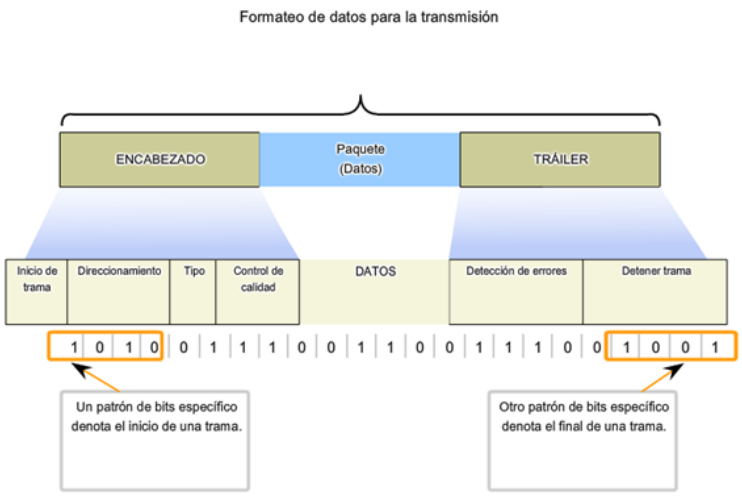


Ilustración 20. Esquema general de un paquete de datos en la capa de red.

RF24NetworkHeader

La clase RF24Network trabaja con la estructura RF24NetworkHeader para implementar las cabeceras de los paquetes, semejantes a la capa de red del modelo OSI (Tmrh20.github.io 2017).

Los métodos más importantes de la clase RF24NetworkHeader son:

- RF24NetworkHeader(): El constructor por defecto. Crea una instancia vacía de una cabecera. Los atributos públicos de la clase son:
  - from_node: Dirección lógica de donde se ha generado el mensaje.
  - to_node: Dirección lógica de destino.
  - id: Identificador secuencia del mensaje. Se incrementa cada vez que se genera un datagrama.
  - type: Tipo de mensaje. Los tipos del 1 al 64 no serán comprobados por la red para verificar su correcta entrega (ACK's). Los mensajes de 65 al 127 sí.

- RF24NetworkHeader(uint16_t _to, unsigned char _type=0): crea una cabecera dirigida a un nodo concreto y con un tipo de mensaje concreto (type: Tipo de mensaje. Los tipos del 1 al 64 no serán comprobados por la red para verificar su correcta entrega (ACK's). Los mensajes de 65 al 127 sí.
- toString(): Nos genera un String con información de la cabecera (id del mensaje, from_node, type). Útil para depuración del código (*debugging*).

Red tipo infraestructura

Nuestro sistema trabaja como una red tipo infraestructura. Se trata de una topología de red en la que los nodos están colocados en forma de árbol. Existe un nodo de enlace troncal, generalmente ocupado por un hub o switch (en nuestro caso el nodo central), desde el que se ramifican los demás nodos. Es una variación de la red en bus, el fallo de un nodo no implica una interrupción en las comunicaciones. Se comparte el mismo canal de comunicaciones.

La topología en árbol puede verse como una combinación de varias topologías en estrella.

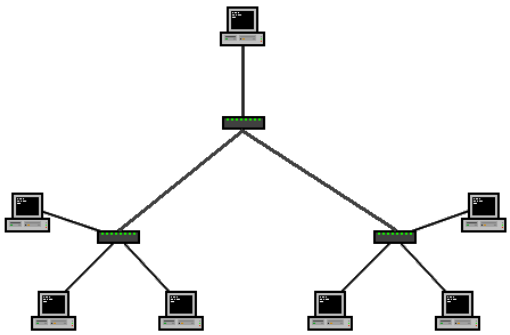


Ilustración 21. Red tipo árbol.

10 El bus SPI (*Serial Peripheral Interface*).

Las comunicaciones entre el módulo Ethernet, el módulo NRF24l01 y el controlador Arduino se realizaran a través del puerto SPI (*Serial Peripheral Interface*). Mediante el SPI los microcontroladores se pueden comunicar con uno o más dispositivos periféricos a cortas distancias. Así mismo también puede ser usado para comunicar dos microcontroladores entre sí.

El bus SPI tiene una arquitectura de tipo maestro-esclavo. El dispositivo maestro (*master*) puede iniciar la comunicación con uno o varios dispositivos esclavos (*slave*), y enviar o recibir datos de ellos. Los dispositivos esclavos no pueden iniciar la comunicación, ni intercambiar datos entre ellos directamente.

En el bus SPI la comunicación de datos entre maestro y esclavos se realiza con dos líneas independientes, una del maestro a los esclavos, y otra de los esclavos al maestro. Por tanto la comunicación es *Full Duplex*, es decir, el maestro puede enviar y recibir datos simultáneamente.

Otra característica de SPI es que es bus síncrono, al contrario que el puerto UART (*Universal Asynchronous Receiver-Transmitter*) como puede ser el puerto USB que es asíncrono. El dispositivo maestro proporciona una señal de reloj, que mantiene a todos los dispositivos sincronizados. Esto reduce la complejidad del sistema frente a los sistemas asíncronos.

Finalmente se requiere una línea adicional SS (*Slave Select*) para cada dispositivo esclavo conectado, que permite seleccionar el dispositivo con el que se va a realizar la comunicación.

Por tanto, el bus SPI requiere un mínimo de 4 líneas:

- MOSI (*Master-out, slave-in*) para la comunicación del maestro al esclavo. El dispositivo maestro utilizará este pin como salida de datos. El, o los, dispositivos esclavos lo utilizarán como entrada de datos.
- MISO (*Master-in, slave-out*) para comunicación del esclavo al maestro. En este caso el dispositivo maestro utilizará este pin como entrada y los esclavos como salida.
- SCK (*Clock)* señal de reloj enviada por el maestro para la sincronización de las lecturas de datos.
- SS (*Slave Select*) señal de selección de esclavo.

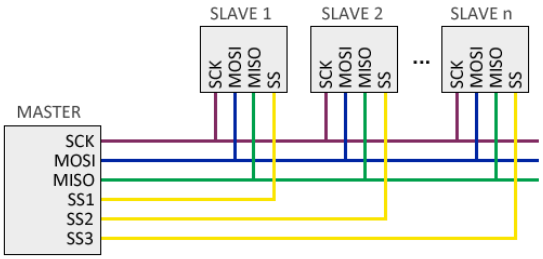


Ilustración 22. Líneas de comunicación SPI.

Por defecto el maestro mantiene en estado HIGH todas las líneas SS. Cuando el maestro quiere establecer comunicación con esclavo pone a LOW la línea SS correspondiente, lo que indica al esclavo que debe iniciar la comunicación.

En cada pulso de la señal de reloj, normalmente en el flanco de subida, el dispositivo maestro envía un bit del esclavo y a la vez puede estar recibiendo un bit del esclavo seleccionado.

La trama (los datos enviados) no sigue ninguna regla, es decir, podemos enviar cualquier secuencia arbitraria de bits. Esto hace que los dispositivos conectados necesiten tener pre-acordado la longitud y significado de los que van a enviar y recibir.

10.1 El bus SPI en Arduino.

Arduino dispone de soporte SPI por hardware vinculado físicamente a ciertos pines. También es posible emplear cualquier otro grupo de pines como bus SPI a través de sofware, pero en ese caso la velocidad será inferior.

Los pines asociados a SPI varían de un modelo a otro. La siguiente tabla muestra la disposición en alguno de los principales modelos.

Modelo	SS	MOSI	MISO	SCK
UNO	10	11	12	13
NANO	10	11	12	13
MINI PRO	10	11	12	13
MEGA	53	51	50	52

Tabla 5. Pines asociados al bus PSI en diferentes placas Arduino.

En placa Arduino como la Mega 2560, entre otras, además de tener los pines mencionados en la tabla para el acceso al bus SPI, tenemos un conjunto de pines denominado ICSP (*In-Circuit Serial Programming*) (Es.wikipedia.org. 2017), compuesto por bloque de 6 pines juntos, o 10 dependiendo del modelo, conectados a los correspondientes pines SPI del microcontrolador.

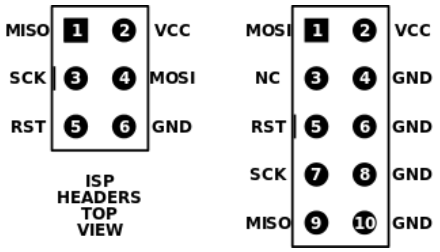


Ilustración 23. Puertos habituales ICSP.

El pin SS por hardware se emplea al usar Arduino como esclavo. En caso de usar Arduino como maestro, podemos usar cualquier pin como SS, o varios en caso de disponer de varios esclavos, como en el caso del nodo central y los módulos Ethernet y NRF24l01.

Para usar el puerto o protocolo SPI en Arduino el IDE Standard (*Integrated Development Environment* - entorno de desarrollo interactivo) proporciona la librería "SPI" que contiene las funciones necesarias para controlar el hardware integrado de SPI.

Asimismo, el entorno de programación de Arduino define las constantes SCK, MOSI, MISO, y SS para los pines de SPI. Usar estos "alias" en nuestro código hace que sea más fácil de intercambiar programas entre modelos placas.

10.2 La librería "SPI".

Esta librería permite comunicar con dispositivos que implementen el protocolo SPI. Antes de empezar una comunicación SPI se deben tener en cuenta una serie de factores que hay que poner en común entre maestro y esclavos:

- Cuál es la máxima velocidad que los dispositivos pueden alcanzar.
- El orden de los bits. Enviamos/recibimos primero el bit más significativo (MSB – *Most Significant Bit*), o primero enviamos/recibimos el menos significativo (LSB – *Least Significant Bit*). La mayoría de los dispositivos trabajan enviando el MSB primero.
- Cuando realizaremos la toma del bit enviado/recibido. Podemos tomar el instante en que se toma la señal recibida en el flanco de subida del reloj o en el flanco de bajada.

Estos tres parámetros pueden ser configurados en la placa Arduino mediante la función SPISettings de la clase SPI disponible en la biblioteca SPI. Es necesario fijar estos parámetros antes de comunicar.

Para la ejecución del presente proyecto no será necesario manejar instrucciones SPI directamente ya que utilizaremos las librerías disponibles para los módulos Ethernet Shield V2 , NRF24L01 y RF24Network, que ya incluyen la utilización de SPI. Con estas librerías, junto con la librería SPI, la comunicación entre el microcontrolador (nuestro Arduino), el micro del Ethernet Shield V2 y el NRF24L01 se realizará de forma transparente para nosotros.

Podemos encontrar una explicación de cada una de las funciones de la librería SPI en la web de Arduino (Arduino.cc 2017).

11 Portal WEB con funciones de gestión y control.

El portal web estará alojado en el sistema central, el servidor WEB se alojará en la memoria de programa del Arudino Mega, donde se albergarán las funciones de gestión y control de sistema.

Servidor con Ethernet Shield V2.

Mediante la clase *EthernetServer*(puerto) crearemos un socket escuchando en el puerto determinado que recibirá peticiones que podrán ser interpretadas y atendidas por programa.

A continuación se muestra un código de ejemplo en el que se pone en funcionamiento un servidor escuchando en el puerto 80 (HTTP), con dirección IP 192.168.1.177, en el que se escuchan las peticiones de clientes HTTP y una vez terminadas las peticiones se responde con una respuesta estándar HTTP seguida de los valores obtenidos de las 6 primeras entrada analógicas de la placa Arduino. En este caso no se realiza ningún procesado de la información solicitada por el cliente (los datos leídos por parte del servidor) y simplemente se responde con la misma respuesta, sea cual sea la petición.

```
/*
  Web Server

 A simple web server that shows the value of the analog input pins.
 using an Arduino Wiznet Ethernet shield.

 Circuit:
```

```
 * Ethernet shield attached to pins 10, 11, 12, 13
 * Analog inputs attached to pins A0 through A5 (optional)

 created 18 Dec 2009
 by David A. Mellis
 modified 9 Apr 2012
 by Tom Igoe

 */

#include <SPI.h>
#include <Ethernet2.h>

// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(192, 168, 1, 177);

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }


  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}


void loop() {
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        // if you've gotten to the end of the line (received a newline
        // character) and the line is blank, the http request has ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");   // the connection will be closed
after completion of the response
          client.println("Refresh: 5");  // refresh the page automatically every 5
sec
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          // output the value of each analog input pin
          for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
            int sensorReading = analogRead(analogChannel);
            client.print("analog input ");
            client.print(analogChannel);
            client.print(" is ");
            client.print(sensorReading);
            client.println("<br />");
          }
          client.println("</html>");
          break;
```

```
        }
        if (c == '\n') {
          // you're starting a new line
          currentLineIsBlank = true;
        }
        else if (c != '\r') {
          // you've gotten a character on the current line
          currentLineIsBlank = false;
        }
      }
    }
    // give the web browser time to receive the data
    delay(1);
    // close the connection:
    client.stop();
    Serial.println("client disconnected");
  }
}
```

Para poder trabajar con el shield Ethernet primero le asignamos una dirección MAC y una IP al módulo mediante la función *Ethernet.begin*(byte mac, IPAddress ip) de manera que el chipset WZ5500 trabaje con estos datos.

Una vez inicializado el módulo Ethernet podemos crear un objeto *EthernetServer* indicando a través de qué puerto TCP estará escuchando. En el ejemplo podemos la línea *EthernetServer server(80)* que crear un objeto *EthernetServer* llamado server y escuchando en el puerto 80.

Con el objeto server creado pasamos a ponerlo en marcha con el método *server.begin()* de manera que comienza a escuchar las peticiones.

Con el método *server.avaiblabe()* obtendremos un objeto de la clase *client* que nos servirá para comunicarnos con el cliente que ha realizado una petición al servidor. La conexión será persistente hasta que se cierre mediante el método *client.stop().*

Mediante el método *client.connected()* podremos obtener el estado de la conexión del cliente. Si un cliente se ha conectado al servidor y a enviado información pero todavía no ha sido leída, el método devolverá el valor *true* como indicativo de que queda información por leer.

El método *client.available()* nos devuelve el número de bytes que hay disponibles para leer, es decir la cantidad de datos que ha enviado el cliente al servidor al que se ha conectado.

El método *client.read()* nos devuelve el siguiente byte disponible en el buffer de datos recibidos en el servidor por parte de el cliente en cuestión. Por cada uso de *client.read()* el buffer se va vaciando y la siguiente llamada al método devolverá el siguiente byte a leer. Leemos byte a byte los datos recibidos y almacenados en el buffer de entrada del módulo Ethernet.

Los métodos *client.print()* y *client.println()* sirven para enviar datos al cliente. Los datos que se envían son cadenas de texto y son enviados carácter a carácter. El método *print* enviada la cadena de texto, el método *println* envía la cadena más un salto de línea.

Por último, el método *client.stop()* cierra la conexión del cliente con el servidor. A partir de este momento no podemos enviar más datos al cliente a no ser que se vuelva a crear un cliente nuevo tras una petición al servidor, mediante el método *server.available()* que devuelve un objeto de la clase *EthernetClient.*

Con el uso del debido código de programa es posible analizar las peticiones del cliente que se conecta al servidor y responder en consecuencia. Este es el método utilizado en el proyecto y puede verse la parte del código utilizado en el nodo central correspondiente al servidor en documentación complementaria.

Uso de *client.print(F(String))* – almacenamiento en memoria FLASH.

Un problema importante con el que nos encontramos en el desarrollo del servidor WEB es la cantidad de memoria necesaria para almacenar el código HTML de las páginas WEB a mostrar. La cantidad de datos necesarios para almacenar el código HTML excede la capacidad de memoria SRAM (*Static Random Access Memory*) del microcontroaldor Mega2560 (8KB).

Las sentencias *client.print()*, al igual que *Serial.print()*, son almacenadas por defecto por el compilador en la memoria SRAM del microcontrolador. Mediante el uso de la directiva F() podemos indicar al compilador que los datos a continuación (cadenas de texto constantes) sean almacenadas en la memoria FLASH (capacidad de 256 KB en Arduino Mega 2560). Se recalca **constantes** porque no es posible almacenar variables en la memoria FLASH.

HTTP, HTML5 y CSS

El servidor WEB trabaja bajo el protocolo HTTP (*Hipertext Transfer Protocol*) para la comunicación con el cliente. Podemos ver información relacionada con HTTP en el anexo nº 2.

Para el desarrollo del diseño de las páginas WEB se utiliza HTML5 (*HyperText Markup Language)* y CSS (*Cascading Stylesheets).* En el anexo nº 3 podemos ver información al respecto.

Para el desarrollo de las funciones de

Autenticación del usuario y Javascript

La implementación del sistema de autenticación se realizará mediante el algoritmo de intercambio de claves *Diffie-Hellman* y el cifrado de los datos de acceso mediante AES 128. Para la realización de estas funciones en el lado cliente (el navegador WEB del PC, tablet o Smartphone) se implementarán funciones en JavaScript que serán enviadas por el servidor a través de HTTP en el código HTML de las páginas WEB.

Se detalla en mayor profundidad el funcionamiento de este aspecto en los anexos nº 4 y nº 8.

12 Obtención de la temperatura.

12.1 El LM35.

El LM35 es un sensor de temperatura digital. A diferencia de otros dispositivos como los termistores en los que la medición de temperatura se obtiene de la medición de su resistencia eléctrica, el LM35 es un integrado con su propio circuito de control, que proporciona una salida de voltaje proporcional a la temperatura.

La salida del LM35 es lineal con la temperatura, incrementando el valor a razón de 10mV por cada grado centígrado. El rango de medición es de -55ºC (-550mV) a 150ºC (1500 mV). Su precisión a temperatura ambiente es de 0,5ºC.

Ilustración 24. Sensor de temperatura LM35DZ.

Sus características más relevantes son:

- Está calibrado directamente en grados Celsius.
- La tensión de salida es proporcional a la temperatura.
- Tiene una precisión garantizada de 0.5 °C a 25 °C.
- Baja impedancia de salida.

- Baja corriente de alimentación (60 µA).
- Bajo coste.

12.2 Esquema de montaje.

A continuación se muestra el conexionado del sensor LM35DZ con el microcontrolador. En este caso se conecta a la entrada analógica A0 (pudiéndose conectar a cualquiera de ellas).

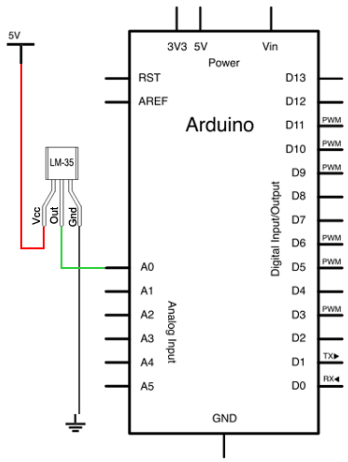


Ilustración 25. Esquema de interconexión LM35DZ.

12.3 Código.

Se muestra a continuación el código de la función para la toma de temperatura. Se realiza la lectura del valor de tensión en la entrada analógica indicada en PIN_LM35 (mediante un *#define* o valor constante por ejemplo) y se transforma dicho valor en función de la relación 1ºC/10mV.

```
float leerTemperatura() {
  /**** Dado que el LM35DZ tiene un rango de valores desde -55ºC (correspndiente a -550m
V) hasta 150ºC (correspondiente a 1500mV), y la temperatura ambiente puede rondar desde
los -10ºC hasta los 45ºC (por ejemplo) tendremos valores entre -10mV y 450 mV.
   *  Por lo tanto podemos modificar el valor de referencia de la tensión analógica del
microcontrolador para que la precisión sea mayor.
   */
  analogReference(INTERNAL1V1);
  uint32_t valorLM35=0;
  for (int i=0;i<10000;i++) {
    valorLM35 += analogRead(PIN_LM35); // Leemos en el pin analógico al que está conecta
do el LM35 10000 veces para sacar un promedio
  }
  valorLM35 = valorLM35/10000; // promedio
  float milivoltios = (1100 /1023) * valorLM35; // Tensión en milivoltios. 1100 son los
milivoltios de la tensión de referencia. 1023 es la resolución del conversor AD (10 bits
 ->1024 valores, del 0 al 1023)
```

```
  float temp=milivoltios/10;
  ////Serial.print("Temperatura local : ");

  ////Serial.println(temp);
  return temp;
}
```

La línea:

```
analogRead(PIN_LM35);
```

Recoge el valor obtenido en la entrada analógica siendo un valor comprendido entre 0 y 1023 ya que la precisión del ADC es de 10 bits. El conversor ADC de Arduino nos dará un valor mapeando un nivel de referencia de tensión asignado mediante la función analogReference().

Arduino permite cambiar la tensión tomada como referencia por el conversor analógico digital. El valor de la referencia se cambia con la función AnalogReference, y los valores posibles son:

- DEFAULT: Valor por defecto, correspondiente con Vcc (5V o 3.3V, según modelos)
- INTERNAL: Corresponde a 1.1V (en Atmega 168 y 328)
- EXTERNAL: Voltaje aplicado de forma externa en el pin Vref (siempre entre 0 y Vcc
- INTERNAL1V1 y INTERNAL2V56, correspondientes a 1.1V y 2.56V (sólo en Mega)

En el caso de usar la referencia de tensión externa (EXTERNAL), si sabemos con total seguridad que una señal no va a superar de un cierto valor de tensión, por ejemplo 0.7V, podemos proporcionar este valor como referencia a través del Pin Aref. La medición se realizará tomando esta tensión como referencia en lugar de Vcc, por lo que es posible ajustar la precisión relativa del conversor AD.

Dado que la tensión que nos proporcionará el sensor LM35DZ podrá oscilar entre -550 mV y 1500 mV (desde -55 ºC hasta 150 ºC), y que las temperaturas ambientales habituales pueden oscilar entre -10 ºC y 45 ºC, tendremos un rango de posibles valores comprendido entre -100 mV y 450mV, por lo que se realizara la modificación de la tensión de referencia del conversor AD para mejorar la precisión de la medida añadiendo la línea:

```
analogReference(INTERNAL1V1); // en arduino Mega.
```

Con esta línea indicamos al conversor AD que el rango de valores posibles de la tensión de entrada oscilará entre 0 y 1.1V de manera que el espacio o resolución entre

valores consecutivos será de 1.1V / 1023 = 1.07 mV, siendo anteriormente de 5V/1023 = 4.88 mV.

13 Información de consumo.

13.1 El ACS712.

El sensor de corriente ACS712 trabaja con un sensor de efecto *Hall* que detecta el campo magnético que se produce por inducción de la corriente que circula por la línea que se está midiendo. EL sensor nos entrega una salida de voltaje proporcional a la corriente, dependiendo la aplicación podemos usar el ACS712-05A, ACS712-20A o el ACS712-30A, para rangos de 5, 20 o 30 amperios respectivamente

El ACS712 podemos encontrarlo en módulos, los cuales nos facilitan sus conexión, traen una bornera para conectar la línea que queremos medir y 3 pines, dos para conectar la alimentación y un pin para la salida analógica.

El rango de corriente que podemos medir y sensibilidad varían dependiendo del modelo del integrado, existen tres modelos:

Modelo	Rango	Sensibilidad
ACS712ELCTR-05B-T	-5 a 5 A	185 mV/A
ACS712ELCTR-20A-T	-20 a 20 A	100 mV/A
ACS712ELCTR-30A-T	-30 a 30 A	66 mV/A

El sensor entrega un valor de 2.5 voltios para una corriente de 0A y a partir de allí incrementa proporcionalmente de acuerdo a la sensibilidad, teniendo una relación lineal entre la salida de voltaje del sensor y la corriente. Dicha relación es una línea recta en una gráfica Voltaje - Corriente donde la pendiente es la sensibilidad y la intersección en el eje Y es 2.5 voltios. La ecuación de la recta es la siguiente:

$$V = m \cdot I + 2.5$$

Por lo tanto la intensidad es:

$$I = \frac{V - 2.5}{m}$$

Para la realización de las medidas de corriente hay que tener en cuenta que las entradas analógicas del microcontralador funcionan como conversores ADC (*Analogic-Digital Converter*) con una resolución dada. La función “analogRead()” nos devolverá un valor entre el rango de resolución del microconotrolador (10 bits en los modelos UNO, Nano, Mega) mapeado en función de una tensión de referencia (por defecto VCC, que normalmente es 5V, o 3.3V).

13.2 Esquema de la instalación.

A continuación se muestra el conexionado del sensor ACS712 con el microcontrolador. En este caso se conecta a la entrada analógica A0 (pudiéndose conectar a cualquiera de ellas).

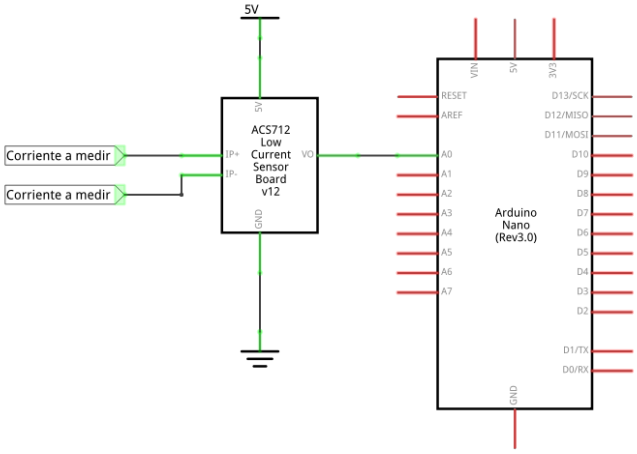


Ilustración 26. Esquema interconexión ACS712.

13.3 Código.

A continuación se muestra código para una toma de medida de corriente en el pin analógico A0 y el cálculo de la potencia activa en Watios.

```
/**** Función Consumo() que nos devolverá el consumo ****/
float Consumo() {

  float Sensibilidad=0.066; //sensibilidad en V/A para nuestro sensor (para el modelo ACS712 ELC 30)
  float offset=0.08; // Equivale a la amplitud del ruido (Obtenida experimentalmente mediante varias mediciones para calibrarlo).

  float voltajeSensor; // Para guardar el voltaje que nos da el sensor.
  byte pin_sensor=0; // Pin analógico donde tenemos la salida del sensor ACS712. En este es el 0.
  float corriente=0; // Para guardar el valor de la corriente (transformándo el valor de la tensión según la relación del fabricante del sensor).
```

```
  long tiempo=millis(); // Para contar el tiempo y realizar varias medidas en un tiempo determinado.
  float Imax=0; // Para guardar el valor máximo de la corriente
  float Imin=0; // Para guardar el valor mínimo de la corriente

  while(millis()-tiempo<50)//realizamos mediciones durante 50 milisegundos. Teniendo en cuenta una red 220 V 50 Hz, en la que cada ciclo dura 20 ms, en 50 ms tomaremos medidas de 2.5 ciclos.
  {
    voltajeSensor = analogRead(pin_sensor) * (5.0 / 1023.0)
    corriente=0.9*corriente+0.1*((voltajeSensor-2.5)/Sensibilidad);   //Ecuación     para obtener la corriente. Es un promedio.
    if(corriente>Imax)Imax=corriente;  // Almacenamos el valor máximo
    if(corriente<Imin)Imin=corriente;  // y mínimo de la corriente
  }

  float  Ip=((Imax-Imin)/2)*1.20-offset;//obtenemos  la  corriente  pico.  El  "1.20"  es  un factor corrector obtenido experimentalmente.
  float Irms=Ip*0.707; //Intensidad RMS (Eficaz) = Ipico/(2^1/2)
  float P=Irms*220.0; // P=IV Watios

  return(P);
}
```

14 Control de la iluminación.

14.1 Los relés como conmutadores o cruzamientos.

Para permitir el control de la iluminación necesitaremos módulos que permitan el actuar sobre la tensión de red que alimenta los puntos de luz en cada una de las estancias. Existen módulos de relés que permiten conmutar la tensión de red de 220 V y 10 A de intensidad máxima (suficiente para puntos de luz) a través de pequeñas señales de tensión continua a 5V que pueden ser proporcionadas por las placas arduino a través de sus salidas digitales. Estos módulos disponen de la electrónica necesaria para activar la bobina del relé. La potencia de salida de las placa Arduino no es suficiente y se podrían dañar por la resonancia producida por la bobina si se conectan directamente.

Ilustración 27. Módulo relé 220 VAC 10A.

El circuito primario de un relé, que recibe la señal de la electrónica de baja tensión, está formado por una bobina arrollada a un núcleo metálico, formando un electroimán.

El circuito secundario, encargado de alimentar la carga, está formado por unos contactos eléctricos instalados en láminas de metal flexible.

Todos los elementos están fijados a una base aislante y rodeados de una envolvente, que impiden que exista el contacto eléctrico entre los distintos terminales o con el exterior.

De estos contactos uno o dos son contactos fijos, mientras que el restante es un contacto móvil encargado de cerrar el circuito con uno de los contactos fijos.

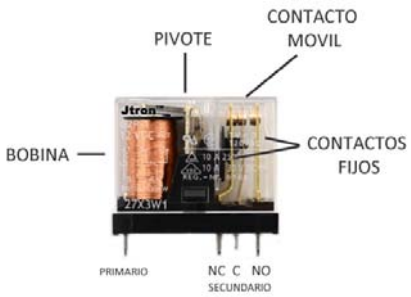


Ilustración 28. Detalle de un relé.

Los relés normalmente disponen de tres contactos en el secundario C (común), NO (normalmente abierto) y NC (normalmente cerrado). Pero también encontramos modelos que prescinden del terminal NC.

14.2 Cambio de interruptores por conmutadores en la vivienda.

La manera de compatibilizar el uso del sistema domótico sobre la iluminación con los actuales sistemas manuales de encendido y apagado de las luces del hogar (interruptores y/o conmutadores) se realizará mediante la interconexión de relés con contactos NO y NC (que actúan como conmutadores) con conmutadores existentes o a instalar (en sustitución de los actuales interruptores).

En las estancias con un único interruptor se deberá sustituir el mismo por un conmutador.

En las estancias en las que existan dos conmutadores, uno de ellos deberá ser sustituido por un cruzamiento.

A continuación se muestran los esquemas de interconexión del relé en las diferentes situaciones (existencia de un único interruptor, o existencia de dos conmutadores).

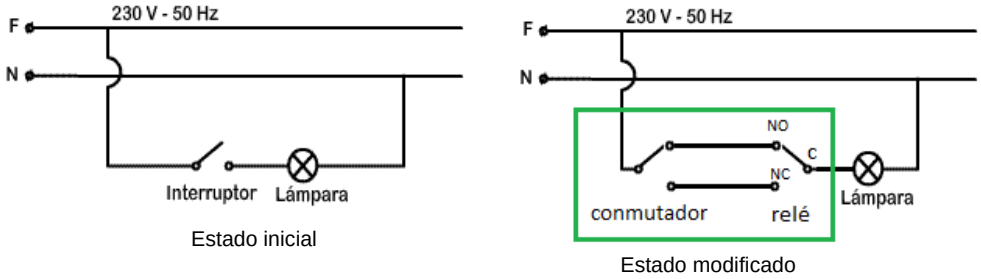


Ilustración 29. Interconexión relé con punto de luz sencillo.

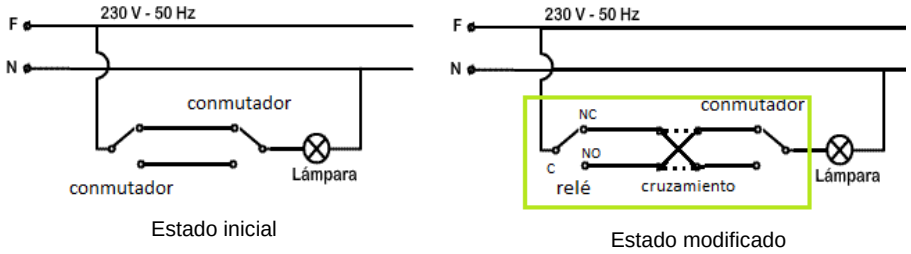


Ilustración 30. Interconexión relé con punto de luz doble.

14.3 Detección del estado de la iluminación con el ACS712.

Para obtener el estado de la iluminación y poder informar al nodo central utilizaremos el módulo ACS712 interpuesto en la línea de alimentación a la lámpara. De este modo estamos midiendo en un momento dado si hay circulación de corriente o no.

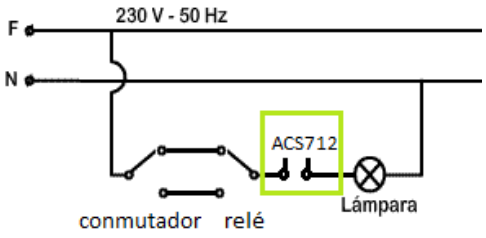


Ilustración 31. Colocación del relé y el sensor de corriente ACS712.

14.4 Esquema de la instalación.

A continuación se presenta el esquema de conexionado del control de iluminación.

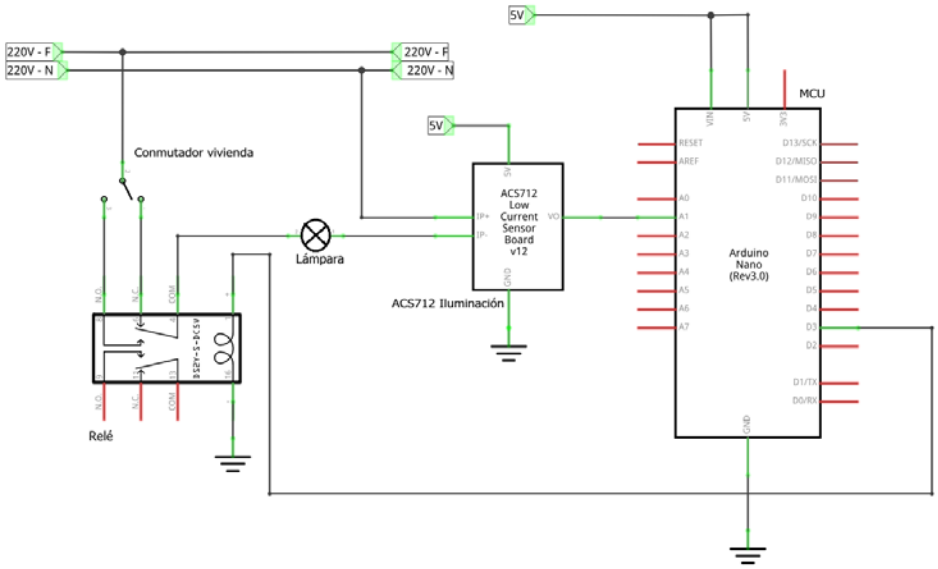


Ilustración 32.Esquema control iluminación.

14.5 Código.

A continuación se muestra código del objeto luz que actuará sobre el control de un punto de luz.

```
/**** Clase Luz que representa el funionamiento de la luz de la estancia ****/
class Luz {
  private:
    boolean Estado; // Representa si la luz está encendida o apgada.
    byte pin_control; // Representa el interruptor. El pin que actuará sobre el relé (pin digital).

  public: // Métodos de la clase Luz.
    Luz(byte pin); // Para crear un objeto luz debemos de indicar sobre que pin debe actuar el microcontrolador.
    void Encender(); // Método que realizará las acciones oportunas para encender la luz.
    void Apagar(); // Método que realizará las acciones oportunas para apagar la luz.
    void setPincontrol(byte pin); // Sirve para fijar o modifcar el pin que se utilizará para gobernar el relé.
    void setEstado(boolean estado); // Para fijar o modificar el valor de la variable Estado.
    boolean getEstado(); // Nos devolverá el valor de la variable Estado.
    byte getPincontrol(); // Nos devolverá el pin de control.

};

/**** Constructor de la clase Luz ****/
Luz::Luz(byte pin) {
  pin_control=pin;
  pinMode(pin_control, OUTPUT); // Configuramos el pin digital de control como OUTPUT.
}

/**** Método que enciende la luz ****/
void Luz::Encender() {
  // Actuar sobre el relé para cambiar el estado solo si la luz está apagada (!Estado).
  Estado=getEstado(); // Almacenamos el estado después antes para comprobar como está la luz.
  if (!Estado) {
```

```
    if (digitalRead(pin_control)) { // Como actuaremos sobre un conmutador debemos cambiar la alimentación actual del relé
       digitalWrite(pin_control,LOW); // Si actualmente estamos activando el relé, dejaremos de activarlo, y viceversa.
    } else {
      digitalWrite(pin_control,HIGH);
    }
  }
}

/**** Método para apagar la luz ****/
void Luz::Apagar() {
  // Actuar sobre el relé para cambiar el estado solo si la luz está encendida (Estado).
  Estado=getEstado(); // Almacenamos el estado antes del cambio para comprobar la luz.
    if (Estado) {
    if (digitalRead(pin_control)) { // Como actuaremos sobre un conmutador debermos cambiar la alimentación actual del relé
       digitalWrite(pin_control,LOW); // Si actualmente estamos activando el relé, dejaremos de activarlo, y viceversa.
    } else {
      digitalWrite(pin_control,HIGH);
    }
  }
}

/**** Método para fijar o modificar el pin de control. Indicamo que el pin de control será una salida del microcontrolador. ****/
void Luz::setPincontrol(byte pin) {pin_control=pin; pinMode(pin_control, OUTPUT);}

/**** Método para fijar el estado de la luz ****/
void Luz::setEstado(boolean estado) {Estado=estado;}

/**** Método para obtener el estado de la luz. Utilizamos el sensor de corriente ACS712 para comprobar el estado de la luz ****/
boolean Luz::getEstado() {
  // Verificar consumo en línea de luz para dar el estado mediante el sensor de corriente
  float voltajeSensor;
  float corriente=0;
  long tiempo=millis();
  float Imax=0;
  float Imin=0;

  while(millis()-tiempo<15)//realizamos mediciones durante 15 milisegundos para asegurarnos del estado de la luz.
  {
    voltajeSensor = analogRead(A0) * (5.0 / 1023.0);//lectura del sensor
    corriente=0.9*corriente+0.1*((voltajeSensor-2.5)/0.066); //Ecuación para obtener la corriente
    if(corriente>Imax)Imax=corriente;
    if(corriente<Imin)Imin=corriente;
  }
  float consumo=(((Imax-Imin)/2)*1.20-0.045);

  if (consumo > 0.1) {Estado=true;} // Si hay consumo la luz está encendida.
  else {Estado=false;} // En caso contrario apagada.

  return Estado;
}

/**** Método para obtener el pin de control ****/
byte Luz::getPincontrol() {return pin_control;}
```

15 Control del estado de las puertas.

El estado de las puertas será controlado mediante contactos magnéticos colocados en las mismas. Se utilizará la versión de los contactos N.A. (Normalmente

Abierto) para reducir el consumo de intensidad que provoca tener el contacto N.C. con la puerta cerrada (situación normal).

15.1 Esquema de la instalación.

A continuación se presenta el esquema de conexión del interruptor magnético con resistencia *pull-down* y el MCU.

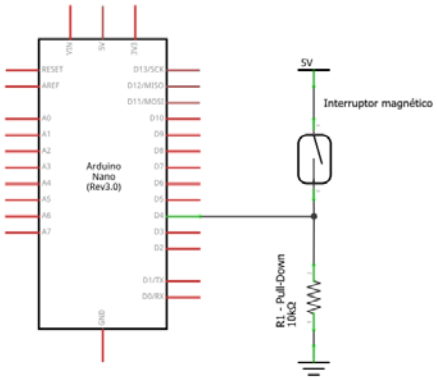


Ilustración 33. Esquema de conexión del interruptor magnético.

15.2 Código.

A continuación se presenta código de una función que devuelve el estado de la puerta conectada la pin digital 4.

```
    boolean verEstadoPuerta() {
      byte pin_sensor = 4; // Pin en el que conectamos el sensor magnético. En este caso el 4 (pin digital).
      pinMode(pin_sensor, INPUT); // Configuramos el pin digital en modo INPUT.
      return(digitalRead(4)); // Leemos el pin digital 4 y devolvemos su estado ("1" o "0").
    }
```

16 Instalación por nodos

Una vez vistos los elementos funcionales que formarán cada nodo y como trabaja cada uno de ellos, podemos pasar a ver el montaje e interconexión de cada nodo y sus elementos.

16.1 Nodo central.

En este apartado se indican los componentes electrónicos que formarán parte del nodo central así como el conexionado de los mismos.

El nodo central estará compuesto por:

- 1 ud. Arduino Mega 2560.
- 1 ud. Ethernet Shield V2 simplificado.
- 1 ud. NRF24l01
- 1 ud. Sensor de temperatura LM35DZ.
- 1 ud. Relés en cascada para actuar sobre persianas.

Podemos conectar varios módulos esclavos a nuestro Arduino mediante el protocolo SPI utilizando varias salidas digitales SS (también llamadas CSN – *Chip Select N*). A continuación se presenta el conexionado del nodo central con el módulo Ethernet V2, el módulo NRF24l01, el sensor LM35DZ y el relé de accionamiento de persianas.

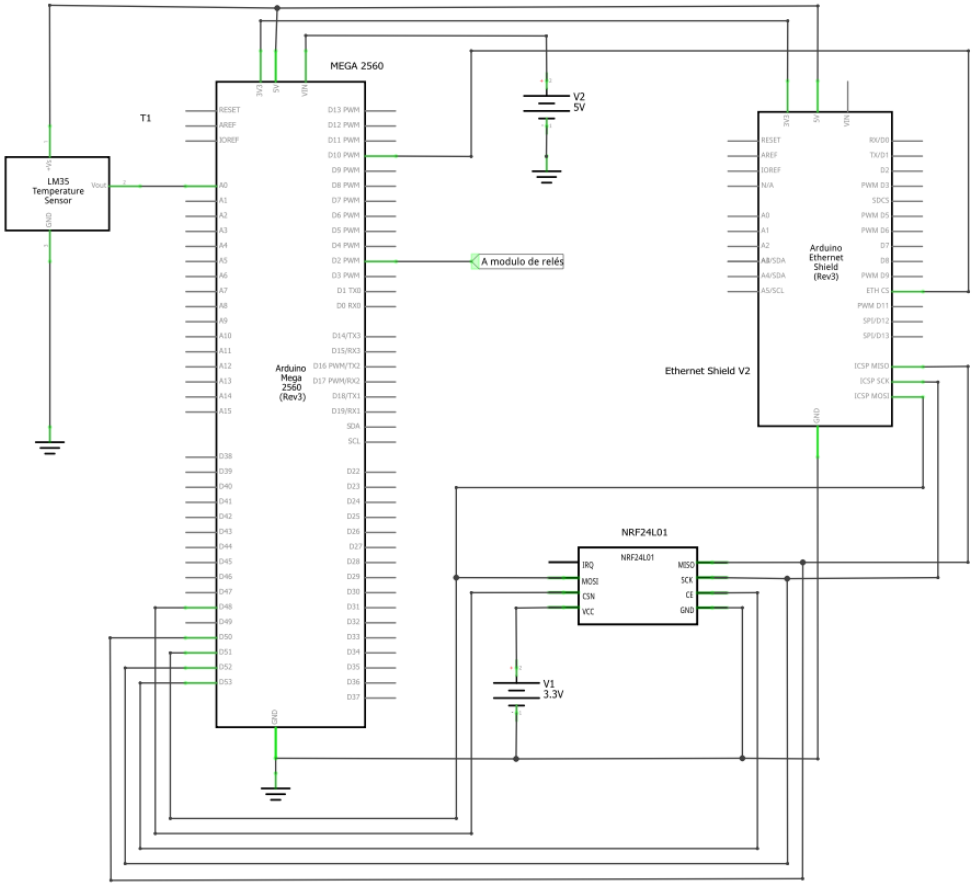


Ilustración 34. Esquema de conexión nodo central.

En el esquema propuesto podemos ver como los puertos de comunicación SPI: MOSI, MISO y SCK (reloj) son comunes a los dos módulos conectados al Arduino Mega que hará de módulo *Master* en la comunicación SPI. El pin de selección de esclavo para el módulo RF (CSN) será el pun 48 de Arduino Mega, y para el módulo Ethernet será el pin 10. Además aparece la conexión del pin CE (*Chip Enable*) del módulo RF al pin número 53 de Arduino Mega 2560 para el control del modo en el transceptor (emisor o receptor).

En el anexo nº 5 podemos ver un diagrama de flujo del funcionamiento del nodo central.

16.2 Nodos remotos.

En este apartado se indican los componentes electrónicos que formarán parte de los nodos remotos así como el conexionado de los mismos. Para ello se presentará un nodo ficticio con todas las funcionalidades del proyecto juntas (iluminación, consumo, persianas, puertas y temperatura). Cada nodo estará formado por los elementos que componen cada función.

Los nodos remotos estarán formados por (ver Ilustración 3):

- 1 ud. Arduino Nano (MCU). Todos los nodos.
- 1 ud. NRF24l01. Todos los nodos.
- 1 ud. Sensor de temperatura LM35DZ. Nodo 022.
- 1 ud. Relés en cascada para actuar sobre persianas. Nodo 031.
- 1 ud. Relé de dos contactos más ACS712 (iluminación). Todos los nodos menos el nodo 011 de control de AACC.
- 1 ud. Contacto magnético NA (puertas). Nodos 03 y 031.
- 1 ud. ACS721 (potencia activa). Nodos 03 y 04.

A continuación se presenta el esquema de conexionado de los diferentes componentes que conforman los nodos remotos. En función del nodo correspondiente se instalarán unos elementos y otros.

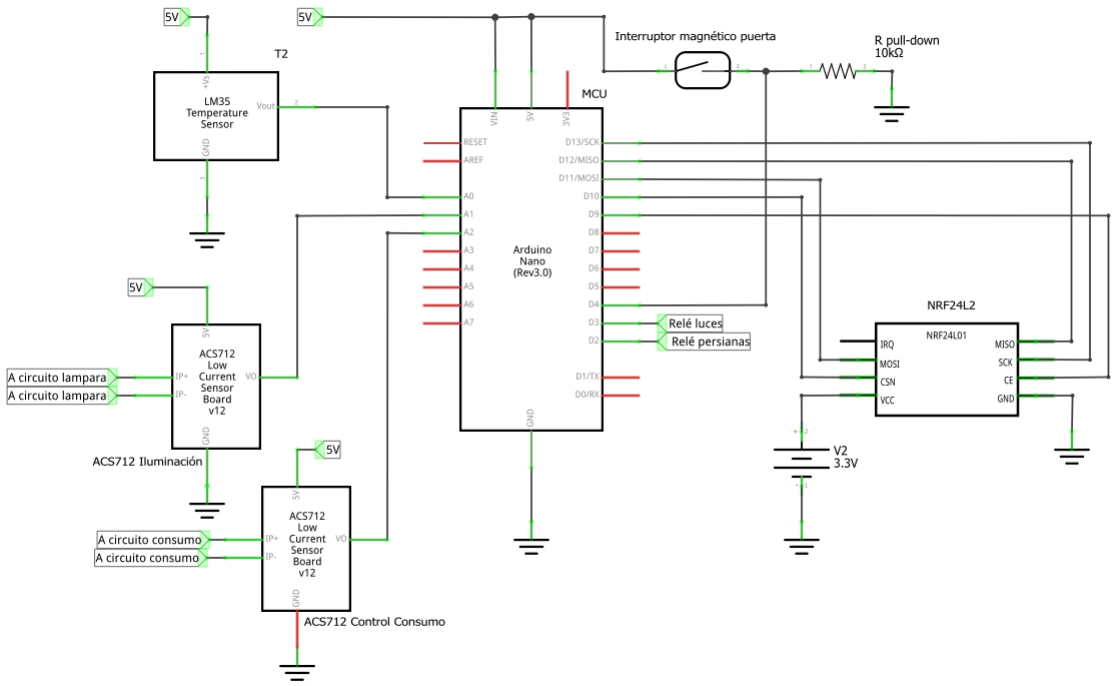


Ilustración 35. Esquema de conexiones en nodos remotos.

Tenemos conectado el sensor de temperatura a la entrada analógica A0, el sensor de corriente para comprobar el estado de la luz en la entrada analógica A1, el sensor de corriente para comprobar el consumo de una línea eléctrica en la entrada A2, el relé para actuar sobre la luz está conectado a la salida digital D3, el relé para actuar sobre las persianas en la salida digital D2 y por último el contacto magnético para el control del estado de la puerta está conectado a la entrada digital D4.

Por otro lado tenemos el módulo NRF24l01 conectado al bus SPI del MCU. La salida digital D9 del MCU actuará como control CE (Chip Enable), la salida digital D10 actuará como CSN (Chip Select), el resto de conexiones SPI se realizan a través del bus establecido en los pines D11 (MOSI), D12 (MISO) y D13 (CLK).

17 Líneas futuras

Tener un sistema accesible desde cualquier lugar a través de internet y con el que poder actuar y/o informarnos del estado de sensores en la vivienda o en la oficina tiene grandes posibilidades. Existen una gran cantidad de ampliaciones y mejoras que se pueden al sistema proyectado.

Algunos ejemplos evidentes pueden ser completar la funcionalidad de control del AACC, o la funcionalidad del sistema de control de persianas.

El control del sistema de aire acondicionado de la vivienda pasa por estudiar el equipo sobre el que hay que actuar ya que una manera sencilla de poder controlarlo puede ser actuar directamente sobre él mediante infrarrojos como si del mando original se tratara.

Existen bibliotecas y ejemplos Arduino en los que es posible captar una secuencia de bits emitidos por un mando a distancia mediante un LED receptor IR (*Infrared Radiation*), guardarla y reproducirla a través de un LED IR emisor. A través de estos ejemplos y las librerías disponibles podremos grabar cada una de las secuencias de cada botón del mando y así poder utilizarlas posteriormente.

Para captar las secuencias IR podremos utilizar el Receptor de IR VS1838B.

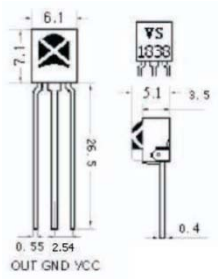


Ilustración 36. Receptor IR VS1838B (encapsulado TO-92)

Y para la emisión IR utilizaremos un diodo IR de los existentes en el mercado.

Ilustración 37. Diodo LED IR.

Para el control de persianas puede ser necesario actuar sobre motores (suponemos que existen motores ya instalados en las persianas) mediante relés de potencia y estos a su vez estarán controlados por otros relés. El motivo por el cual utilizaremos un par de relés, uno de mayor potencia que el otro, es que no podemos atacar directamente el relé de alta potencia con la señal de Arduino porque, además de no tener suficiente potencia de salida, podemos dañar el microcontrolador debido a las oscilaciones que produce la bobina del relé. Los motores de las persianas generan picos de corriente en los arranques que puede estropear o reducir el tiempo de vida de

los relés de 220 VAC 10 A, por lo que se propone el uso de relés en cascada de manera que el relé que actúe sobre el motor soporte una carga de 30 A como mínimo.

Ilustración 38. Relé de alta potencia.

Para el control del estado de las persianas podríamos usar una medición del tiempo de subida o bajada de la persiana realizando unas medidas de los tiempos que tarda el motor. Otra alternativa podría ser utilizar un potenciómetro lineal como el mostrado en la Ilustración 39 colocado en el cajón de la persiana que variará su resistencia en función de la cantidad de persiana enrollada en el eje. El potenciómetro estará en contacto con la persiana enrollada sobre el eje de manera que en función de la cantidad de persiana que haya recogida nos dará una resistencia que podremos utilizar en un divisor de tensión. La tensión de este divisor será variable en función del estado de la persiana y esta tensión podrá ser tomada por el micro controlador Arduino para interpretar la situación.

Ilustración 39. Potenciómetro línea para medir posición.

También podemos desarrollar el contenido de las páginas WEB en archivos almacenados en una tarjeta microSD y mediante el lector de tarjetas incorporado en los módulos Ethernet Shield, o con módulo independiente, podemos acceder a dichos archivos y enviarlos al cliente WEB.

El sistema admite la instalación de memorias FLASH externas tipo SD, micro SD, etc. mediante módulos acoplables a la placa central. En la ilustración nº 4 podemos ver un ejemplo de módulo para memorias tipo micro SD.

Ilustración 40. Lector micro-SD Arduino

18 Conclusiones

Tras la experiencia en la elaboración del proyecto y la implementación del prototipo he conocido un sistema de código abierto y hardware libre con un gran abanico de posibilidades. Pero hay mucho más detrás de esta memoria.

He investigado y aprendido de un modo práctico el funcionamiento de las comunicaciones HTTP, el lenguaje HTML, las hojas de estilo en cascada CSS, el lenguaje JavaScript, el funcionamiento de un sistema de comunicaciones RF basado en GFSK, esta comunicación dirigida mediante el direccionamiento, el funcionamiento de un ISP y la IP pública, el funcionamiento de un servicio DDNS, modificar la configuración del router para habilitar el *port-forwarding*, he comprendido e implementado el algoritmo de intercambio de claves *Diffie-Hellman,* el algoritmo de cifrado AES.

He conocido plataformas de desarrollo colaborativo como GitHub dónde se pueden encontrar gran cantidad de proyectos y librerías para Arduino y muchas más plataformas.

He implementado el prototipo y he podido estudiar su funcionamiento y me he dado cuenta de las posibilidades de mejora y ampliación que existen.

Por lo que puedo decir que el proyecto ha concluido con éxito personal y además puede llegar a ser un proyecto funcional, tras un proceso de producto final, para ser puesto en el mercado o como proyecto a compartir en una comunidad colaborativa.

19 Referencias

Arduino.cc. (2016a). *Arduino - Ethernet*. [online] Available at: https://www.arduino.cc/en/Reference/Ethernet [Consultado Marzo. 2017].

Arduino.cc. (2016b). *Arduino - Introduction*. [online] Available at: https://www.arduino.cc/en/Guide/Introduction [Consultado marzo. 2017].

Arduino.cc. (2017). *Arduino - SPI*. [online] Available at: https://www.arduino.cc/en/reference/SPI [Consultado Marzo. 2017].

EducaChip. (2016). *Arduino Ethernet Shield - Controla Tu Casa Por Internet*. [online] Available at: http://www.educachip.com/arduino-ethernet-shield/ [Consultado Marzo. 2017].

Electrónica Embajadores, (2017). [online] Available at: https://www.electronicaembajadores.com/es/Subfamilias/Productos/CJPP/cajas-convencionales-de-plastico [Consultado Junio. 2017].

Es.wikipedia.org. (2016a). *Cultura Maker*. [online] Available at: https://es.wikipedia.org/wiki/Cultura_Maker [Consultado Marzo. 2017].

Es.wikipedia.org. (2016b). *Serial Peripheral Interface*. [online] Available at: https://es.wikipedia.org/wiki/Serial_Peripheral_Interface [Consultado 6 Abril. 2017].

Es.wikipedia.org. (2016c). *Internet de las cosas*. [online] Available at: https://es.wikipedia.org/wiki/Internet_de_las_cosas [Consultado 6 Marzo. 2017].

Es.wikipedia.org. (2017). *Programación en el sistema*. [online] Available at: https://es.wikipedia.org/wiki/Programaci%C3%B3n_en_el_sistema [Consultado Abril. 2017].

En.wikipedia.org. (2017). *Z-Wave*. [online] Available at: https://en.wikipedia.org/wiki/Z-Wave [Consultado Mayo. 2017].

Forum.arduino.cc. (2016). *Actualizar un DDNS desde Arduino*. [online] Available at: http://forum.arduino.cc/index.php?topic=161191.0 [Consultado Mayo. 2017].

GitHub. (2017). *nRF24/RF24Network*. [online] Available at: https://github.com/nRF24/RF24Network [Consultado Abril. 2017].

Instructables.com. (2017). [online] Available at: http://www.instructables.com/id/Ethernet-Switching-with-Arduino/ [Consultado Marzo. 2017].

Landoni, B. (2016). *Arduino DDNS (Dynamic DNS)*. [online] Open Electronics. Available at: http://www.open-electronics.org/arduino-ddns-dynamic-dns/ [Consultado Abril. 2017].

Luis Llamas. (2017a). *El bus SPI en Arduino*. [online] Available at: https://www.luisllamas.es/arduino-spi/ [Consultado Abril. 2017].

Luis Llamas. (2017b). *Medir intensidad y consumo eléctrico con Arduino y ACS712*. [online] Available at: https://www.luisllamas.es/arduino-intensidad-consumo-electrico-acs712/ [Consultado Mayo. 2017].

Luis Llamas. (2017c). *Medir temperatura con Arduino y sensor LM35*. [online] Available at: https://www.luisllamas.es/medir-temperatura-con-arduino-y-sensor-lm35/ [Consultado Mayo. 2017].

María, J. (2016). *DNS Dinamico con ddclient (Dynamic DNS) | Trasteando con Arduino*. [online] Trasteandoarduino.com. Available at: https://trasteandoarduino.com/2016/03/17/dns-dinamico-dynamic-dns-con-ddclient/ [Consultado Mayo. 2017].

Noip.com. (2017). *Free Dynamic DNS - Managed DNS - Managed Email - Domain Registration - No-IP*. [online] Available at: https://www.noip.com/ [Consultado Junio. 2017].

Processing.org. (2017). *Processing.org*. [online] Available at: https://processing.org/ [Consultado Marzo. 2017].

Prometec.net. (2016). *Comunicación dúplex con NRF2401 | Tutoriales Arduino*. [online] Available at: http://www.prometec.net/duplex-nrf2401 [Consultado Abril. 2017].

Real Academia Española. (2017). Domótica. En *Diccionario de la lengua española* (23ª ed.). Recuperado de http://dle.rae.es/?id=E7W0v9b

Real Decreto 346/2011, de 11 de marzo, por el que aprueba el Reglamento regulador de las infraestructuras comunes de telecomunicaciones para el acceso a los

servicios de telecomunicaciones en el interior de las edificaciones. *Boletín Oficial del Estado.* Madrid, 1 de abril del 2011, núm. 78, p. 33816.

Real Decreto 842/2002, de 2 de agosto, por el que se aprueba el Reglamento Electrotécnico para Baja Tensión. *Boletín Oficial del Estado.* Madrid, 18 de septiembre del 2002, Suplemento del BOE núm. 224. Reglamente Electrotécnico para Baja Tensión. Guía técnica de aplicación Instalaciones de sistemas de automatización, gestión técnica de la energía y seguridad para viviendas y edificios (GUÍA-BT- 51) (Ed. 2007).

Tmrh20.github.io. (2017). *Newly Optimized RF24Network Layer: RF24NetworkHeader Struct Reference*. [online] Available at: http://tmrh20.github.io/RF24Network/structRF24NetworkHeader.html [Consultado Abril. 2017].

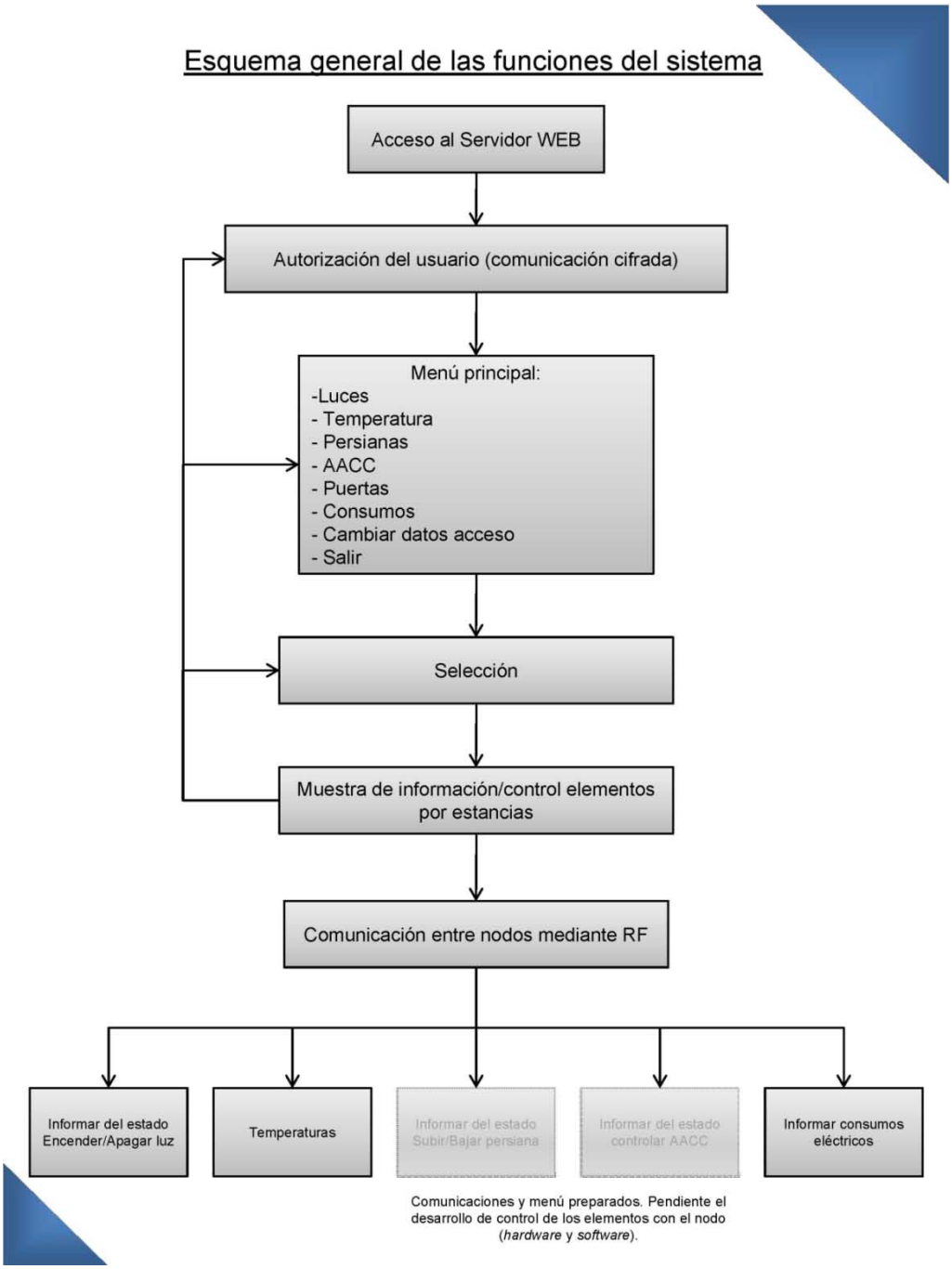
Wiring.org.co. (2016). *Wiring*. [online] Available at: http://wiring.org.co/ [Consultado Marzo. 2017].

Wiznet.co.kr. (2016). *W5500 Ethernet Shield | WIZnet Co., Ltd.*. [online] Available at: http://www.wiznet.co.kr/product-item/w5500-ethernet-shield/ [Consultado Marzo. 2017].

Zwave.es. (2017). *Tienda domótica online: Sistemas de domóticos inalámbricos z-wave*. [online] Available at: http://zwave.es [Consultado Junio. 2017].

ANEXOS

ANEXO 1 - Esquema general de las funciones del sistema.



ANEXO 2 – Comunicación HTTP.

Una vez hemos obtenido un método mediante el cual poder transmitir datos sobre el protocolo TCP/IP desde el sistema hacia el exterior o recibir peticiones desde el exterior hacia el sistema (actuando como servidor), podremos pasar a utilizar este flujo de datos para comunicarnos con navegadores WEB mediante el uso de HTTP a través del puerto 80.

HTTP (*Hipertext Transfer Protocol*) es el protocolo de comunicación que permite las transferencias de información en la WEB cuya especificación se encuentra reflejada en el RFC 2616 (versión 1.1 de HTTP). HTTP es un protocolo sin estado, lo que quiere decir que no guarda información sobre conexiones anteriores. El desarrollo de aplicaciones WEB necesita frecuentemente mantener estado y para ello se utilizan las *cookies*.

HTTP sigue un esquema de petición-respuesta entre un cliente y un servidor. El cliente, que suele ser llamado "agente de usuario" o "*user agent"*, realiza una petición enviando un mensaje, con cierto formato estándar al servidor. El servidor responderá con un mensaje de respuesta.

Mensajes HTTP:

Los mensajes HTTP son en texto plano y tiene la siguiente estructura general:

- Línea inicial (termina con retorno de carro y salto de línea). En las peticiones (de cliente a servidor) el formato de la línea es 'método de petición' seguido de la URL del recurso y la versión HTTP que soporta el cliente. Para las respuestas (de servidor a cliente) el formato de la línea es 'versión del HTTP usado' seguido del código de respuesta y un mensaje asociado a dicha respuesta.
- Las cabeceras del mensaje que terminan con una línea en blanco y pueden estar formadas por varias líneas. Son metadatos y facilitan información útil a los servidores y clientes (navegadores WEB en general).
- Por último el cuerpo del mensaje.

Métodos de petición HTTP

El protocolo define una serie de métodos de petición y cada uno de ellos indica la acción que desea que se efectué sobre el recurso solicitado.

- HEAD: Pide una respuesta idéntica a la que correspondería a una petición GET, pero en la petición no se devuelve el cuerpo del mensaje. Sirve para recuperar metadatos de encabezados.
- GET: Pide una respuesta sobre el recurso especificado. El servidor, normalmente, responderá con el recurso solicitado en el cuerpo del mensaje de respuesta, o con alguna acción realizada en el servidor.
- POST: Envía los datos para que sean procesados por el recurso identificado. Los datos se incluirán en el cuerpo de la petición.
- PUT: Sube o carga un recurso especificado en el servidor.
- DELETE: Borra un recurso especificado.
- TRACE: Solicita al servidor que en la respuesta meta todos los datos que reciba en el mensaje de petición. Se utiliza con fines de depuración y diagnóstico ya que el cliente puede ver lo que llega al servidor y de esta forma ver lo que añaden al mensaje los servidores intermedios.
- OPTIONS: Devuelve los métodos HTTP que el servidor soporta para un URL específico. Esto puede ser utilizado para comprobar la funcionalidad de un servidor web mediante petición en lugar de un recurso específico.
- CONNECT: Se utiliza para saber si se tiene acceso a un host, no necesariamente la petición llega al servidor, este método se utiliza principalmente para saber si un proxy nos da acceso a un host bajo condiciones especiales, como por ejemplo "corrientes" de datos bidireccionales encriptadas (como lo requiere SSL).
- ….

Hay una más cantidad de métodos de petición que se utilizan en el protocolo HTTP, pero que nuestro sistema no va a implementar, por lo que no se considera necesario mayor información sobre los métodos de petición.

Las respuestas a estos métodos de petición han de ser implementados mediante código de programa ya que no existe ninguna librería que tenga implementadas las respuestas como servidor WEB según HTTP (sí existe la librería o clase HttpClient que implementa ciertas peticiones HTTP de las expuestas aquí)

Códigos de respuesta:

Las respuestas HTTP del servidor comienzan con un código de respuesta seguidas del resto de contenido asociado a la respuesta. Cada código tiene un significado concreto y el cliente lo recibirá y procesará en consecuencia. Los códigos habituales de respuesta son:

- 1xx: Respuestas informativas. Indica que la petición ha sido recibida y se está procesando.
- 2xx: Respuestas correctas. Indican que la petición ha sido procesada correctamente.
- 3xx: Respuestas de redirección. Indica que el cliente necesita realizar más acciones para finalizar la petición.
- 4xx: Errores causados por el cliente. Indica que ha habido un error en el procesado de la petición a causa de un fallo en el servidor.
- 5xx: Errores causados por el servidor. Indica que ha habido un error en el procesado de la petición a causa de un fallo en el servidor.

Cabeceras HTTP:

Son los metadatos que se envían en las peticiones o respuestas HTTP para proporcionar información esencial sobre la transacción en curso. Cada cabecera es especificada por un nombre de cabecera seguido por dos puntos, un espacio en blanco y el valor de dicha cabecera seguida de un retorno de carro por un salto de línea. Una línea en blanco señala el final de las cabeceras y el comienzo del cuerpo del mensaje en su caso. Si no hay cabeceras la línea en blanco debe permanecer igualmente para indicar el paso al mensaje o la finalización de una petición.

Podemos clasificar las cabeceras según su función:

- Cabeceras que indican las capacidades aceptadas por el que envía el mensaje: *Accept* (indica el MIME aceptado), *Accept-Charset* (indica el código de caracteres aceptado), *Accept-Encoding* (indica el método de compresión aceptado), *Accept-Language* (indica el idioma aceptado), *User-Agent* (para describir al cliente), *Server* (indica el tipo de servidor), *Allow* (métodos permitidos para el recurso).
- Cabeceras que describen el contenido: *Content-Type* (indica el MIME del contenido), *Content-Length* (longitud del mensaje), *Content-Range*, *Content-Encoding, Content-Language, Content-Location.*
- Cabeceras que hacen referencias a URIs: *Location* (indica donde está el contenido), *Referer* (Indica el origen de la petición).
- Cabeceras que permiten ahorrar transmisiones: *Date* (fecha de creación), *If-Modified-Since*, *If-Unmodified-Since, If-Match, If-None-Match, If-Range, Expires, Last-Modified, Cache-Control, Via, Pragma, Etag, Age, Retry-After.*
- Cabeceras para control de cookies: *Set-Cookie, Cookie*

- Cabeceras para autentificación: *Authorization, WW-Authenticate*
- Cabeceras para describir la comunicación: *Host* (indica máquina destino del mensaje), *Connection* (indica cómo establecer la conexión)

A continuación podemos ver un ejemplo de comunicación HTTP dónde se pueden apreciar los métodos de petición, las caberas, el código de respuesta y el cuerpo del mensaje.

Petición:

```
GET /index.html HTTP/1.1
 Host: www.example.com
 User-Agent: nombre-cliente
 Referer: www.google.com
 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101
Firefox/45.0
 Connection: keep-alive
 [Línea en blanco]
```

Respuesta:

```
HTTP/1.1 200 OK
Date: Fri, 31 Dec 2003 23:59:59 GMT
Content-Type: text/html
Content-Length: 1221

<html lang="eo">
<head>
<meta charset="utf-8">
<title>Título del sitio</title>
</head>
<body>
<h1>Página principal de tuHost</h1>
(Contenido)
  .
  .
  .
</body>
</html>
```

ANEXO 3 - HTML5 y CSS

HTML

Con el protocolo de transferencia de Hipertexto (HTTP) podemos efectuar la comunicación con los navegadores WEB al tratarse de un estándar de uso en internet, lo que nos brinda la posibilidad de comunicar nuestro sistema con cualquier dispositivo con acceso a internet y un navegador WEB.

Para mostrar información en los navegadores WEB es preciso enviar información que estos entiendan y procesen para mostrar los resultados por pantalla. El uso del lenguaje de programación de páginas WEB HTML (*HyperText Markup Language)* nos permite realizar la interfaz gráfica y de comandos que el navegador WEB del dispositivo que se conecte a nuestro sistema mostrará por pantalla.

El HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>,/). El HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir o hacer referencia a un tipo de programa llamado script, el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

HTML también sirve para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (como HTML 4.01 y anteriores).

HTML consta de varios componentes vitales, entre ellos los elementos y sus atributos, tipos de data y la declaración de tipo de documento.

No es objeto del TFG aprender a programa HTML, pero sí el ver las posibilidades que tenemos al usar HTML para presentar nuestras páginas WEB a los navegadores.

HMTL5 y CSS

Con HTML tenemos un lenguaje de programación sencillo para generar las presentaciones de las páginas WEB que el servidor mostrará en el navegador que se conecte a él. Sin embargo, el avance del lenguaje de programación HTML ha dado lugar a nuevos lenguajes mejorados, como el lenguaje HTML5, que permite

representar pequeñas entidades gráficas que dotan a las páginas WEB de más posibilidades.

Otro elemento importante a tener en cuenta a la hora de programar una página WEB son las hojas de estilo en cascada (CSS - *Cascading Stylesheets)*. Se trata de un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en lenguaje marcado. Se utiliza generalmente para establecer el diseño visual de las páginas WEB, e interfaces de usuario escritas en HTML o XHTML. Junto con HTML y JavaScript, CSS es una tecnología usada por muchos sitios web para crear páginas visualmente atractivas, interfaces de usuario para aplicaciones web, y GUIs para muchas aplicaciones móviles.

ANEXO 4 - Javascript

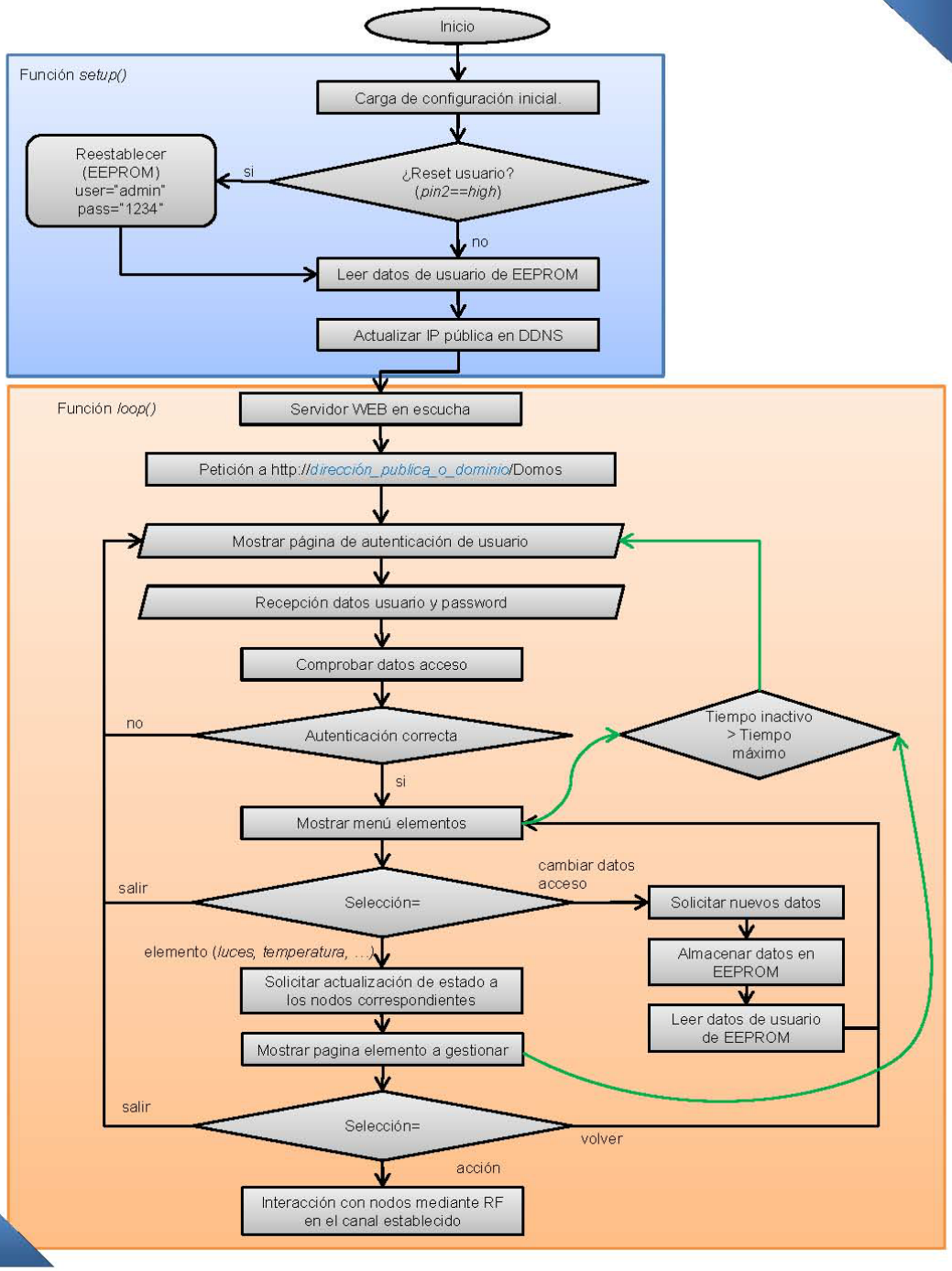
Abreviado JS es un lenguaje de programación interpretado y orientado a objetos. Comúnmente utilizado en el lado del cliente (código enviado por el servidor al cliente para su posterior ejecución). El navegador WEB ha de soportar JS para poder ejecutar los códigos de programa que envía el servidor.

El uso más común de JavaScript es escribir funciones embebidas o incluidas en páginas HTML y que interactúan con el Document Object Model (DOM o Modelo de Objetos del Documento) de la página. Algunos ejemplos sencillos de este uso son:

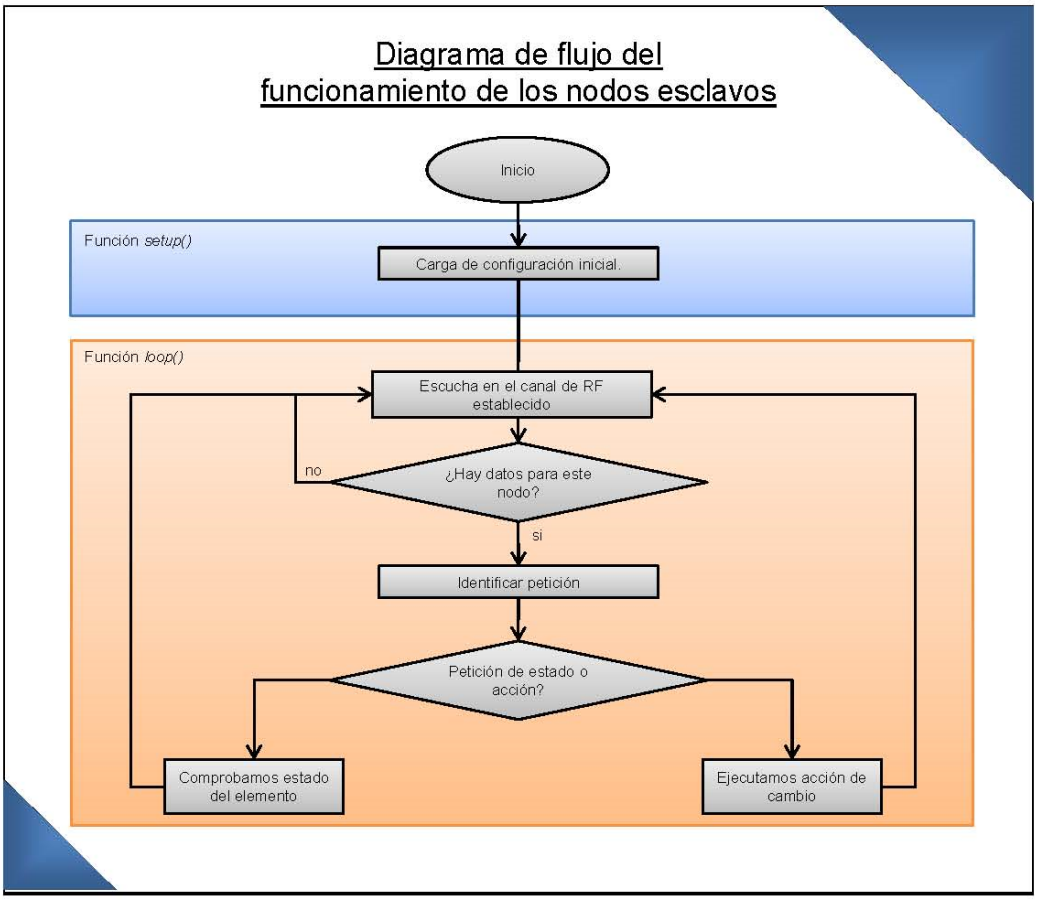
- Cargar nuevo contenido para la página o enviar datos al servidor a través de AJAX sin necesidad de recargar la página (por ejemplo, una red social puede permitir al usuario enviar actualizaciones de estado sin salir de la página).
- Animación de los elementos de página, hacerlos desaparecer, cambiar su tamaño, moverlos, etc.
- Contenido interactivo, por ejemplo, juegos y reproducción de audio y vídeo.
- Validación de los valores de entrada de un formulario web para asegurarse de que son aceptables antes de ser enviado al servidor.
- Transmisión de información sobre los hábitos de lectura de los usuarios y las actividades de navegación a varios sitios web. Las páginas Web con frecuencia lo hacen para hacer análisis web, seguimiento de anuncios, la personalización o para otros fines.

Mediante JavaScript crearemos una función que se ejecutará en el navegador del cliente a través de la cual mediante unos números aleatorios generados en el servidor la información de usuario y contraseña viajará cifrada con una función simple. El procesador de los microcontroladores de Arduino utilizados en TFG no son capaces de trabajar con HTTPS.

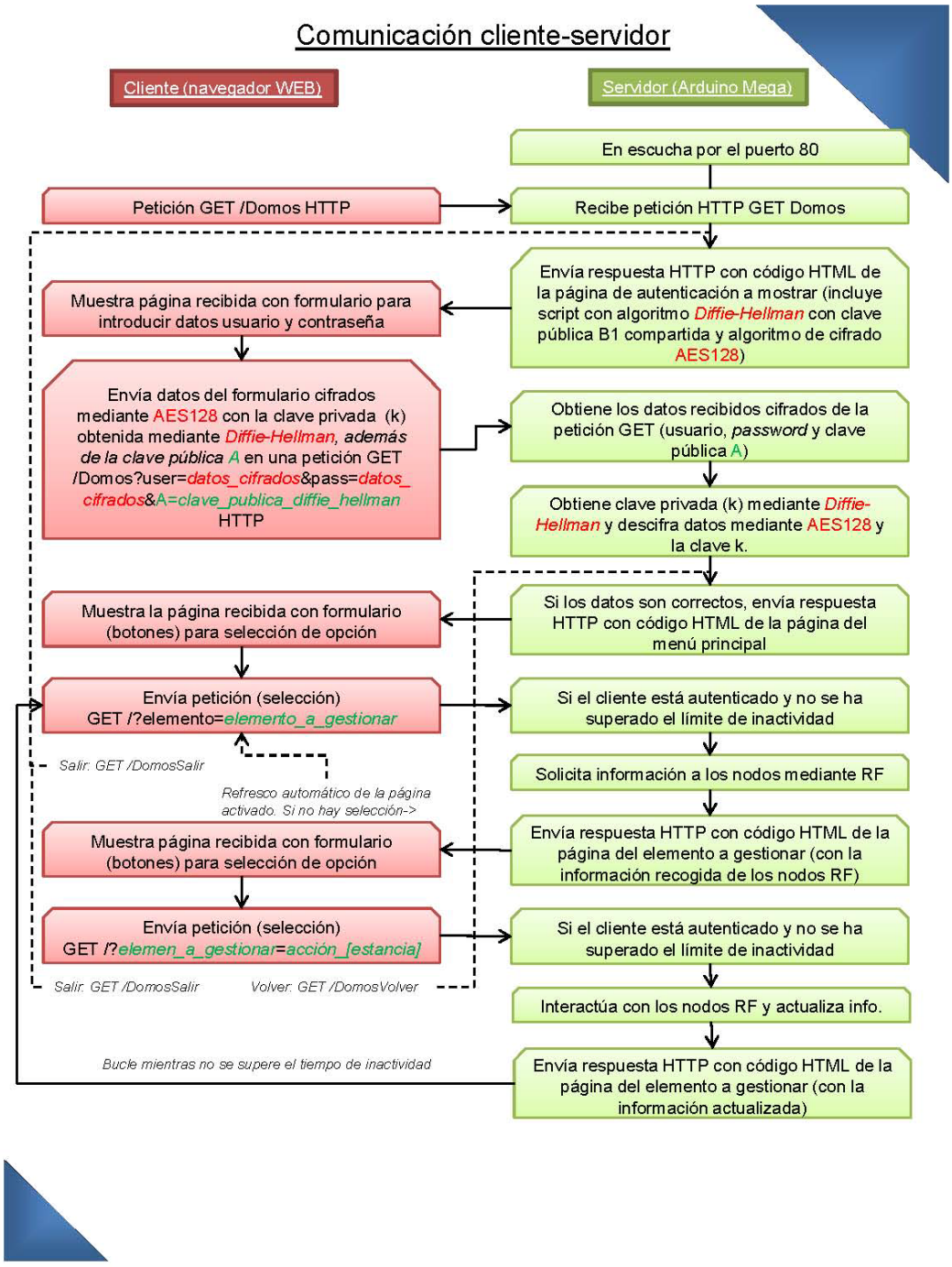
ANEXO 5 – Diagrama de flujo del nodo central



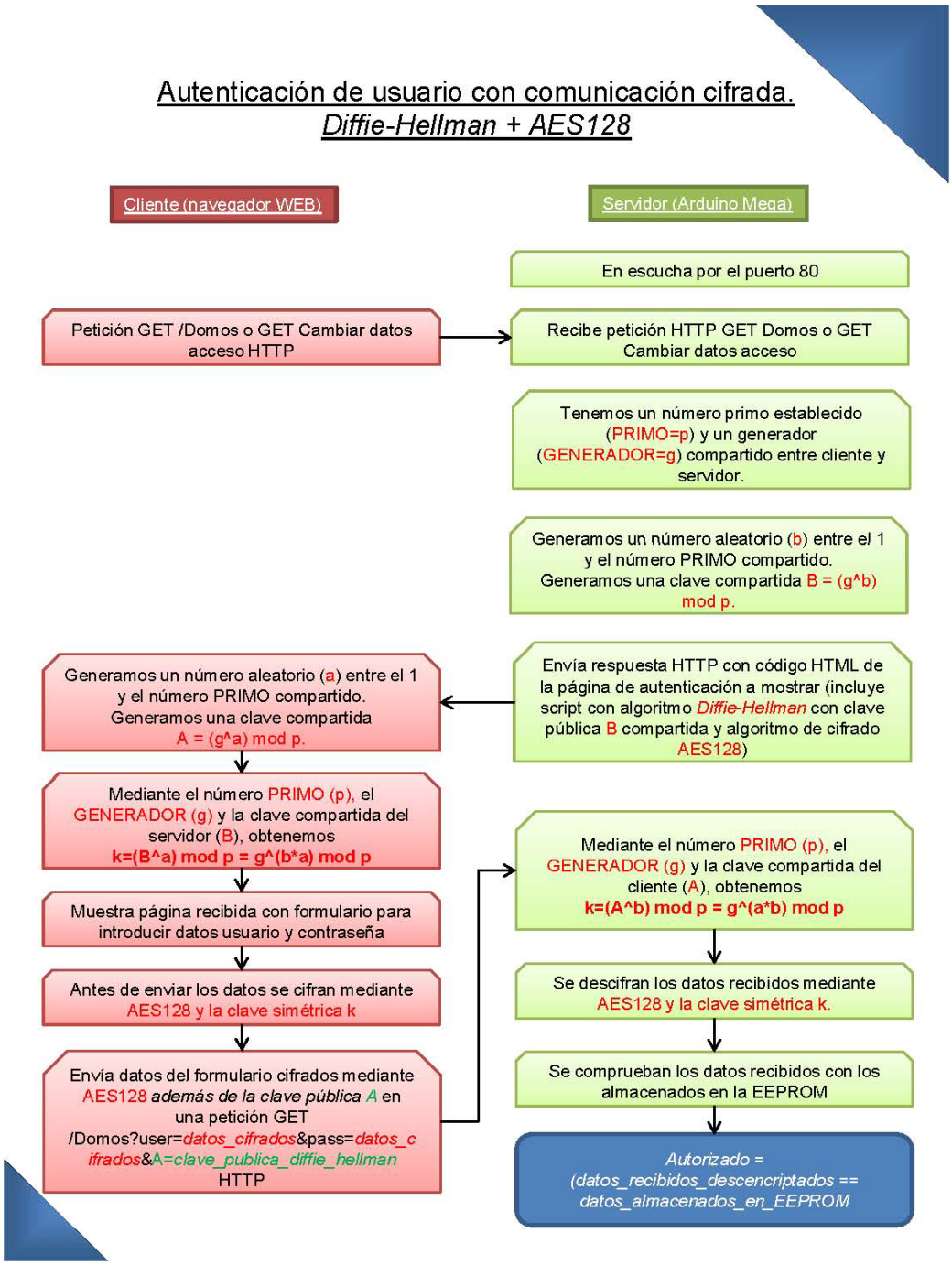
ANEXO 6 – Diagrama de flujo del nodo central



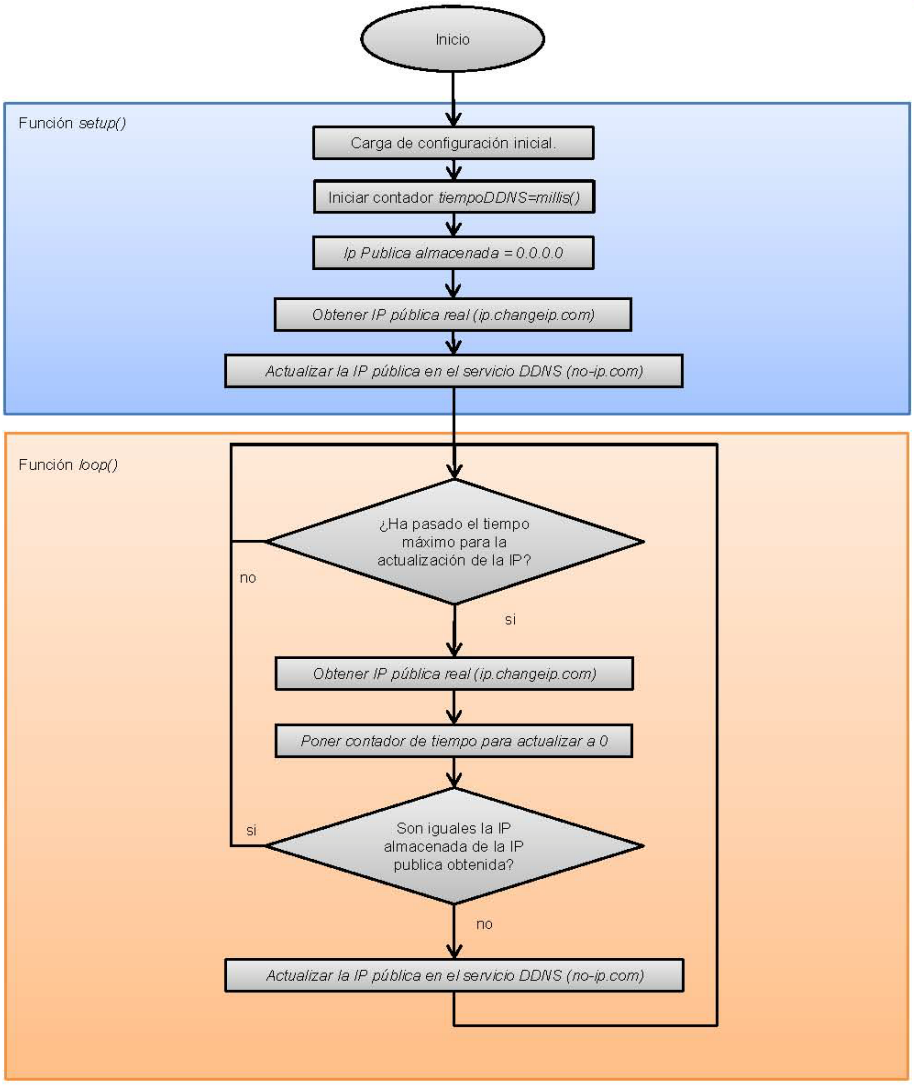
ANEXO 7 – Diagrama comunicación cliente-servidor



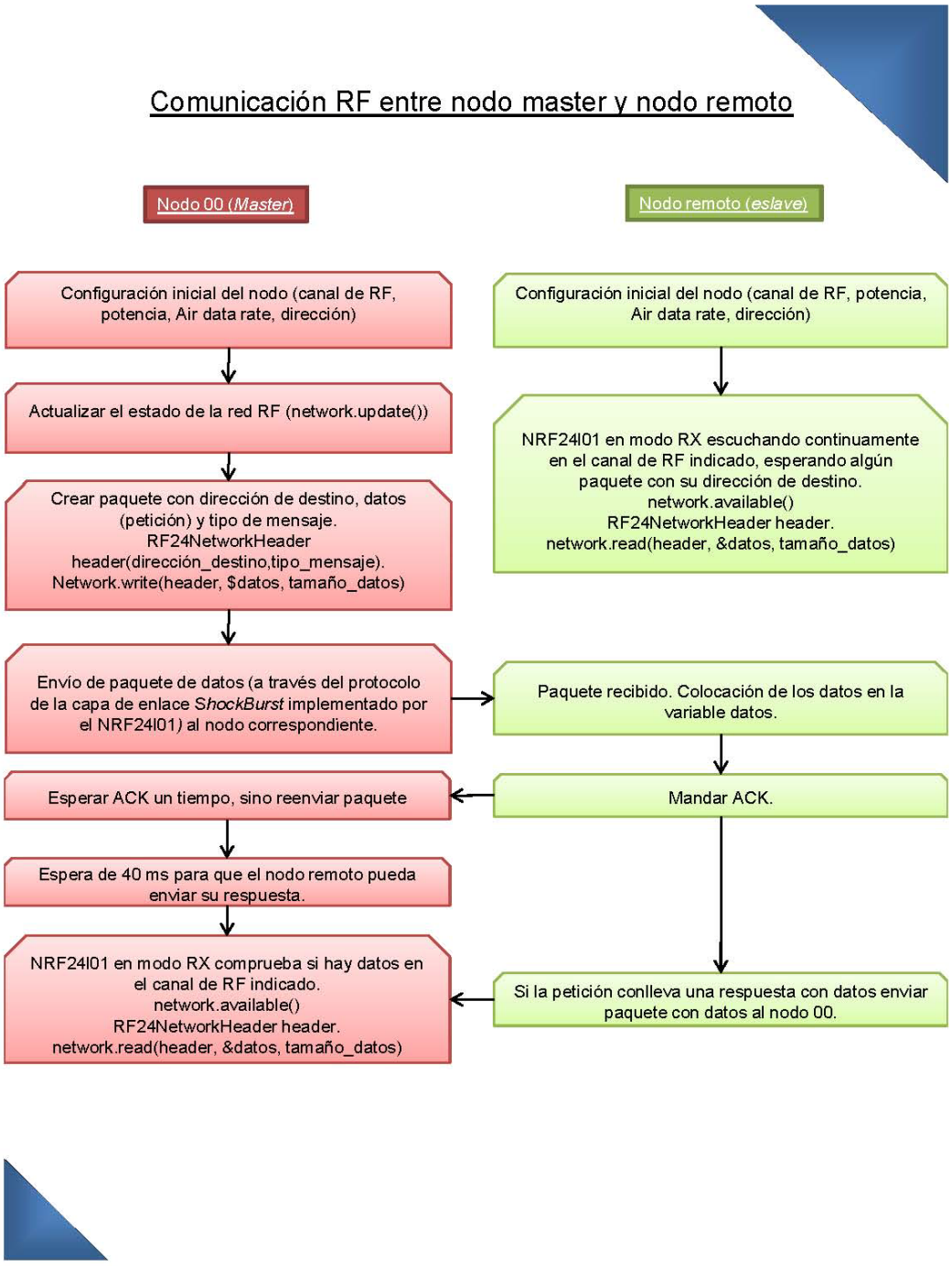
ANEXO 8 – Diagrama autenticación usuario



ANEXO 9 – Diagrama actualización IP pública en DDNS



ANEXO 10 – Comunicación RF Nodo master-nodos remotos



Documentación complementaria

En formato digital se adjuntan:

- Código para nodo central “Nodo_00_v_25.ino”
- Código para nodos remotos “Nodos_remotos_v_8.ino”
- El archivo “Diagramas de funcionamiento.pdf” con los diagramas mostrados en anexos.
- Librería RF24Network (RF24Network-master.zip)
- Planificación.
- Especificaciones técnicas:
  - ACS712
  - LM35DZ
  - nRF24L01
  - W5500