UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Trabajo Final de Grado

Desarrollo de un sistema de monitorización domiciliaria basado en la plataforma NodeMCU V3

Autor: Aitor Carricondo Monter

Director: David Cuesta Frau

Escuela Politécnica Superior de Alcoi

Grado en Ingeniería Informática

Septiembre 2017

Agradecimientos

Primero y principal, me gustaría dar las gracias a...

- Toda mi familia y, en especial, a mi tía, por toda la ayuda que me han brindado para poder cursar mis estudios, tanto, económica, como moral.
- A mi novia por todo el apoyo que me ha brindado.
- A la Universidad Politécnica de Valéncia por la oportunidad de estudiar en una de las mejores Universidades del país.
- Por último, a todos los profesores que he tenido en mi vida, que me han ayudado a llegar a este momento.
- GRACIAS A TODOS!

Declaración de Autoría

Aquí declaro, que soy el autor de este trabajo. Incluyendo el código fuente, así como, el montaje de la maqueta y la realización de esta memoria. He utilizado fuentes de información que citaré en el apartado final de bibliografía.

Lugar, Fecha	Firma

Resumen

Este TFG tratará sobre el terreno de la domótica, es decir, el conjunto de tecnologías que permiten controlar y/o monitorizar muchos aspectos del hogar, como, por ejemplo, el control de las luces, de persianas, de la temperatura, etc...

En este caso, el trabajo se basa en la construcción de una maqueta de una casa sobre la que, utilizando una placa arduino-based, en concreto la NodeMCU V3 LoLin, poder controlar diferentes sensores y actuadores, como sensores de temperatura, servomotores, sensores de distancia, etc...

Debido a la conectividad Wi-Fi de la que dispone esta placa, para poder monitorizar o controlar los diferentes sensores y actuadores se puede programar un servidor web en la misma placa. Para acceder al servidor web que se crea, la placa se puede configurar como punto de acceso, o, conectarse a una red Wi-Fi de la que se conozcan el SSID y la contraseña, opción, esta útlima que es la que se va a utilizar en este caso.

En concreto, la maqueta tiene, cocina, garaje, habitación, jardín delantero y jardín trasero. En el jardín delantero, o entrada a la casa, dispone de un interruptor magnético en la puerta de acceso principal, para saber si esta abierta o cerrada, así como, un sensor de distancia en la puerta del garaje, para saber si hay algún vehículo cerca de la misma.

Por tanto, el trabajo a realizar es el siguiente: -Montaje de la maqueta y los sensores sobre esta. -Programación del servidor web en la microcontroladora NodeMCU. -Programación del sketch necesario para el control de los sensores y actuadores sobre el servidor web. -Grabación y Edición de un vídeo que muestre el funcionamiento de la maqueta, ya que, debido a su tamaño, sería difícil de transportar. -Redacción de la presentación y la memoria para este Trabajo de Final de Grado.

Índice general

1 Introducción

En esta memoria voy a intentar explicar detalladamente el procedimiento que he seguido en la realización de mi trabajo de final de grado, en el cual, he realizado un proyecto domótico, que consiste en la automatización y control de diferentes partes de una casa, oficina, almacén, o cualquier espacio. En concreto, mi trabajo se basa en la monitorización y control de diferentes partes de una típica casa familiar, como serían, las luces, puertas, control de temperatura, etc...

Para conseguir dicho propósito, he utilizado una placa de desarrollo **Open-Hardware** poco conocida, almenos en territorio español, la **NodeMCU** LoLin, en su versión tercera (V3).

Esta placa de desarrollo esta basada en el SoC ESP8266, el cual dispone de varios pines de propósito general, también llamados pines GPIO, los cuales utilizo para controlar diferentes sensores y actuadores, mostrando su estado y las diferentes opciones en cada uno de estos sensores o actuadores en una página webalojada en la misma placa, ya que, mediante software se puede crear un servidor web que corra en la CPU de la placa y controlar desde ahí los pines GPIO.

1.1. Motivación del Trabajo

Mi motivación principal para elegir un trabajo de estas características, fué mi interés en el mundo de la domótica, es decir, los diversos sistemas que pueden existir para automatizar procesos del día a día de una vivienda, oficina, o, en definitiva cualquier tipo de edificio. Además, este proyecto también se encuadraría dentro del mundo del **IoT**, o, Internet of Things, ya que, se trataría de conectar a Ïnternet"diferentes objetos o sistemas cotidianos, campo, este, que se prevé que va a tener un crecimiento muy grande en los próximos años, cosa, que ya se puede observar en la actualidad, ya que, existen cantidad de tipos de objetos diferentes que se venden ahora mismo, que disponen de conexión a internet, véase, bombillas inteligentes, todo tipo de electrodomésticos con conexión a internet, (lavadoras, lavavajillas, neveras, etc...).

Otra de las razones, fué por el hecho de poder utilizar los conocimientos que podía adquirir al realizar este trabajo, para realizar proyectos de domótica low-cost en mi propia casa, así como, intentar mejorar mis aptitudes y técnicas de programación.

1.2. Open-Hardware

El concepto **Open-Hardware** o Hardware Libre, nace de forma paralela al de Open-Source o Software Libre, y, se basa en los mismos principios de este, es decir, la libertad de uso, de estudio y modificación, de distribución, y de redistribución de las versiones modificadas, a su vez, también existen diferentes licencias de uso y distribución como ocurre con el caso del software.

Los ejemplos más reconocibles de Hardware Libre, son **Arduino** y **Raspberry Pi**, siendo estas dos iniciativas las más conocidas y reconocidas de todo el movimiento Hardware Libre, aunque, existen también otros proyectos muy interesantes, tales como, la impresora 3D autorreplicable RepRap, la cual, esta diseñada de forma que, partiendo de una impresora 3D funcional, se pueden imprimir las piezas necesarias para montar otra impresora igual, todo esto, gracias al uso de un **Arduino** Mega como microcontrolador.

Por lo que respecta a este trabajo, se basa única y exclusivamente en Software y Hardware Libre, ya que, se hace uso de Software Libre con la utilización de código **Arduino** y uso de Hardware Libre gracias al uso de **NodeMCU**.

1.3. Inmótica

La inmótica es el conjunto de tecnologías aplicadas al control y la automatización inteligente de edificios no destinados a vivienda, ya sean, hoteles, oficinas, almacenes y todo tipo de edificios terciarios, aportando así diferentes beneficios a estos edificios, tales como, la mejora en la gestión energética, la mejora en la seguridad y el confort, así como, la mejora de la comunicación entre el usuario y el sistema.

Los equipos y sistemas de automatización y control de edificios se encargan de proporcionar diferentes funciones a un edificio, como pueden ser, la gestión de la calefacción, de la ventilación, del agua caliente, de la iluminación, entre otras funciones. La mayor ventaja de utilizar este tipo de sistemas automatizados en un edificio como una oficina, es la posibilidad de programar la mayoría de elementos para que funcionen el tiempo del día en el que el edificio vaya a estar ocupado y mejorar así la gestión energética.

Las funciones que tienen efecto sobre la eficiencia energética de los edificios se dividen en tres grupos: funciones de regulación automática, funciones de sistemas de automatización y control de edificios y funciones de gestión técnica de edificios.

* **Funciones de regulación automática**
  - Regulación de calefacción y refrigeración.
  - Regulación de la ventilación y el aire acondicionado.
  - Control de iluminación.
  - Control de persianas.
* **Automatización y control de edificios**
  - Adaptación centralizada del sistema de automatización a las necesidades del usuario.
  - Optimización centralizada del sistema de automatización de edificios.
* **Gestión técnica de edificios con funciones de eficiencia energética**
  - Detección de fallos de los edificios y de sus sistemas técnicos y prestación de soporte para el diagnóstico de estos fallos.
  - Presentación de la información sobre consumo de energía, condiciones interiores y posibilidades de mejora.

Normalmente, la utilización de sistemas inmóticos, conduce a una mejora notable en la eficiencia energética de los edificios, contribuyendo así a reducir las emisiones de gases de efecto hinvernadero. La automatización de los equipos de control proporciona la oportunidad de ahorrar energía, comparado con el control manual por parte de los usuarios, viéndose este ahorro incrementado, si se tienen en cuenta los funciones de control más complejas y integradas.

1.4. Domótica

La domótica es la tecnología basada en sistemas capaces de automatizar cualquier tipo de edificación, ya sea, una vivienda, una oficina o un almacén, aportando utilidades tales como, la gestión energética, la seguridad, el bienestar o la comunicación, y que pueden estar integrados mediante redes interiores o exteriores de comunicación, ya sean, cableadas o inalámbricas, y el control de dichos sistemas se caracteriza por una cierta ubicuidad, es decir, sistemas que se pueden controlar, desde dentro o desde fuera de la edificación en cuestión. Una forma de resumirlo sería, sistemas que integran la tecnología en el diseño inteligente de un recinto cerrado.

Los diferentes servicios o beneficios que puede aportar la domótica se dividen en cinco grupos.

- **Ahorro Energético:** Este puede ser el mayor beneficio de un sistema domótico, ya que, permite, mediante diferentes sensores o contadores, la optimización del uso de la energia, por ejemplo, a la hora de encender la climatización de una vivienda, nos ofrece la oportunidad de programar el encendido o apagado, la temperatura que queremos obtener, o que zonas de la casa queremos climatizar. Otros ejemplos serían, el control de toldos o persianas eléctricas, la gestión eléctrica de los diferentes aparatos o electrodomésticos de los que dispone una vivienda o oficina, o el uso de energías renovables.

- **Confort:** El confort o comodidad, conlleva todas aquellas actuaciones que se puedan llevar a cabo para aumentar el grado de comodidad o satisfacción en una vivienda, tanto de carácter activo, como pasivo. Por ejemplo, el control de la iluminación, la automatización de diferentes instalaciones para su fácil control o manejo o la integración del control de entrada o salida de la vivienda.

- **Seguridad:** Otro de los apartados importantes de la domótica, es la mejora de la seguridad en una vivienda, ya que, permite proteger tanto los bienes materiales como la seguridad personal de una familia, mediante sistemas tales como, los sistemas de alarmas de intrusión, los detectores y alarmas de incendios, los escapes de agua o filtraciones, además de, diferentes sistemas de alerta médica o teleasistencia o el acceso a cámaras IP.

- **Comunicaciones:** Se trata de todos los sistemas de comunicaciones de los que dispone un hogar, es decir, todo tipo de controles remotos para los diferentes electrodomésticos, la teleasistencia, el telemantenimiento, los informes de consumo energético y sus costes, la transmisión de alarmas, o, sistemas como los videoporteros.

- **Accesibilidad:** Otro de los enfoques de la domótica, se centra en la mejora de las viviendas para personas con limitaciones físicas o psicológicas, para este fin, existe un concepto llamado, diseño para todos, que no es más que, un movimiento que pretende inculcar la necesidad de que los diseños de productos o servicios tengan en cuenta las necesidades de todos los posibles usuarios, incluyendo las personas con diferentes limitaciones.

Uno de los caminos que parece que va a tomar la domótica, es, la inclusión de diferentes tipos de asistentes inteligentes para controlar los sistemas domóticos, como por ejemplo, el caso de Apple con su asistente Siri y la tecnología Homekit, Samsung con Bixby y Samsung Smart Home, Amazon con su línea de productos que integran su asistente Alexa o el gigante Google con su sistema Google Now junto con Google Home.

1.5. Protocolos de Comunicación

Partiendo de la premisa de que no existe un protocolo estándar de comunicación entre todas las secciones de un sistema domótico, si existen varios protocolos muy extendidos, robustos y fiables, entre ellos, los más utilizados por fabricantes e instaladores son, el protocolo **KNX**, el protocolo **X10** o el **ZigBee**. Paso a detallar cada uno de ellos.

1.5.1. Protocolo KNX

Se trata de un estándar de protocolo de comunicaciones de red, basado en OSI, para edificios inteligentes, este estándar es la convergencia de tres estándares previos, el European Home Systems Protocol (EHS), el European Installation Bus (EIB) y el BatiBus. El estándar **KNX** está gestionado por la asociación homónima **KNX** y el texto de la especificación **KNX** se adquiere previo pago de una tarifa.

Estos tres estándares previos, intentaron desarrollar sus mercados separadamente, pero, en 1997, los tres consorcios decidieron unir sus fuerzas para desarrollar el mercado del hogar inteligente, acordando crear una norma industrial común para poder proponerla como norma internacional. En 2002 fue presentada la especificación **KNX**, logrando entrar lentamente en el mercado, a pesar de ser un sistema bastante robusto y fiable.

KNX define varios medios de comunicación física, tales como, el cableado de par trenzado, la comunicación por red eléctrica, la comunicación mediante radiofrecuencia o el ethernet.

La especificación **KNX** consta de 4 grupos de elementos.

* **Actuadores:** Los actuadores son los elementos del sistema que se conectan físicamente con los elementos a controlar en un edificio, como por ejemplo, las luces, los motores, las válvulas o electroválvulas, estos actuadores se encargan de traducir las instrucciones recibidas del mundo **KNX** al mundo físico.

* **Sensores:** Los sensores son los elementos del sistema que recogen los datos del entorno o interpretan las órdenes del usuario, ejemplos de sensores serían, los pulsadores, los detectores de movimiento o el termostato.

* **Pasarelas:** Las pasarelas enlazan otros sistemas con otros protocolos de comunicación a **KNX**, estos equipos permiten interaccionar con proyectores, otros sistemas inteligentes o comunicarse en remoto con nuestro sistema.

* **Acopladores:** Los acopladores realizan una separación física dentro del bus consiguiendo agrupar los dispositivos en un segmento de características determinadas para la cantidad de equipos, ubicaciones físicas o funciones determinadas y conectarlos con otro segmento para una mayor eficacia en el envío de datagramas a través del bus o alcanzar mayores distancias (Repetidores).

El futuro próximo de esta especificación, parece ser que pasa por la encapsulación de las comunicaciones dentro del protocolo IP, con la especificación **KNX**/IP, pudiendo así conectar los dispositivos compatibles con el estándar **KNX** a las redes que la mayoría de viviendas y oficinas tienen ya instaladas, es decir, las redes TCP/IP con cableado UTP.

1.5.2. Protocolo ZigBee

Se trata de una especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radiodifusión digital de bajo consumo, se basa en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (WPAN). El objetivo de esta especificación es, el de conseguir una comunicación segura para bajas tasas de envío de datos y la maximización de la vida útil de las baterías de los equipos que la usan.

Las principales características de los diferentes aparatos que hacen uso de esta especificación son:

* El bajo consumo.
* La topología de la red que forman los dispositivos, topología de red en malla.
* La facilidad de la integración de los dispositivos, debido a la posibilidad de fabricar nodos con muy poca electrónica

ZigBee hace uso de la banda ISM para todo tipo de usos, en concreto, la banda de 868 MHz en Europa, la de 915 MHz en Estados Unidos y la 2.4 GHz en todo el mundo, aunque, la mayoría de fabricantes que utilizan esta especificación, optan por desarrollar sus dispositivos para la banda de 2.4 GHz, ya que, es una banda libre en todo el mundo. El desarrollo de la tecnología se centra en la sencillez y el bajo costo, más que, por ejemplo, otras redes inalámbricas de la familia WPAN como es el caso del Bluetooth. Otra de las características de **ZigBee**, es, el hecho de que en los nodos más complicados, necesita un 10 % del hardware que pueden necesitar los nodos para Bluetooth o Wi-Fi, bajando esta cifra hasta el 2 % en los nodos más sencillos, no obstante, el tamaño del código necesario representa el 50 % del código de un nodo Bluetooth.

Se suele comparar esta tecnología al Bluetooth, aunque, **ZigBee** tiene varias diferencias y ventajas con la misma.

* Una red **ZigBee** puede tener hasta 65535 nodos, distribuidos en 255 subredes, por los 8 máximos de una red Bluetooth.

* Tiene un menor consumo eléctrico que Bluetooth, exactamente, **ZigBee** consume 30 mA transmitiendo y 3 μA en reposo, frente a los 40 mA transmitiendo y 0.2 mA en reposo de los dispositivos Bluetooth.

* Tiene una velocidad de hasta 250 kbit/s, mientras que, en la tecnología Bluetooth es de hasta 3000 kbit/s.

* Debido a esta diferencia de velocidad, estas tecnologías tienen un uso diferente, mientras que Bluetooth se usa para los teléfonos móviles y la informática casera, **ZigBee** es apropiado para la domótica, los productos dependientes de batería, los sensores médicos y en artículos de juguetería, donde la transferencia de datos es menor.

En la especificación **ZigBee** se definen tres tipos distintos de dispositivos según su papel en la red, estos pueden ser.

* **Coordinador ZigBee (ZigBee Coordinator, ZC):** Son el tipo de dispositivo **ZigBee** más completo. Debe existir al menos un coordinador por red **ZigBee**. Sus funciones son las de encargarse de controlar la red y los caminos que deben seguir los dispositivos para conectarse entre ellos.

* **Router ZigBee (ZigBee Router, ZR):** Este router interconectan los distintos dispositivos repartidos por la topología de la red, además de ofrecer un nivel de aplicación para la ejecución de código de usuario.

* **Dispositivo Final (ZigBee End Device, ZED):** Poseen la funcionalidad de comunicarse con su nodo padre (Coordinador o Router), pero no puede transmitir información destinada a otros dispositivos finales. De esta forma, este tipo de nodo puede estar dormido la mayor parte del tiempo, aumentando la vida media de su batería. Estos son los más baratos.

Como ejemplo para ilustrar el funcionamiento de esta tecnología, se puede imaginar una habitación, donde disponemos de varios ZED's, tales como, un interruptor o una lámpara, y una red de interconexión realizada con routers **ZigBee** y un coordinador **ZigBee**.

Al mismo tiempo, básandose en la funcionalidad, se puede plantear una segunda clasificación de dispositivos.

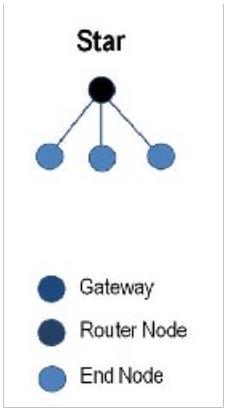
* **Dispositivo de Funcionalidad Completa (FFD):** También se conocen como, nodos activos. Son capaces de recibir mensajes en formato 802.15.4. Gracias a la memoria adicional y a la capacidad de cómputo, pueden funcionar como un coordinador o router **ZigBee**, o pueden ser usados en dispositivos de red que actúen de interfaz con los usuarios.

* **Dispositivo de Funcionalidad Reducida (RFD):** Conocidos como nodos pasivos, tienen capacidad y funcionalidad limitadas con el objetivo de conseguir un bajo coste y simplicidad. Básicamente, serían los sensores y actuadores de la red.

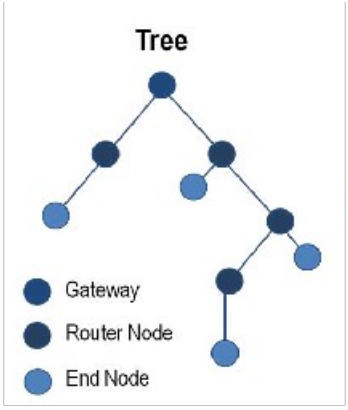
El bajo consumo que se puede conseguir usando esta tecnología, se debe al hecho de, que, un nodo **ZigBee**, ya sea activo o pasivo, puede permanecer dormido la mayor parte del tiempo, es decir, sin consumir energía, pero, cuando se requiere su uso, despierta en unos 15 ms.

Otro de los puntos fuertes de **ZigBee** es, el hecho de permitir tres topologías de red diferentes:

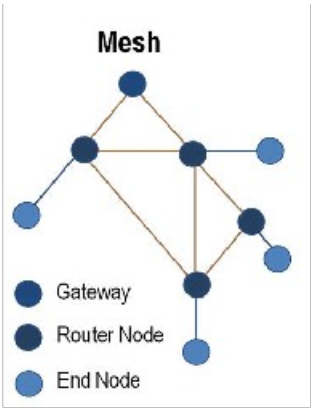
- Topología en estrella: El coordinador se sitúa en el centro.



- Topología en árbol: El coordinador será la raíz del árbol.



– Topología en malla: Al menos uno de los nodos tendrá más de dos conexiones.



La topología más interesante, y, una de las razones por las que puede triunfar esta tecnología, es la topología en malla, ya que, permite que, si un nodo falla, la comunicación puede seguir entre todos los demás nodos debido a que se rehacen los caminos para llegar al coordinador.

En las redes **ZigBee** se pueden usar dos tipos de entornos o sistemas.

* **Con Balizas**

  Es un mecanismo de control del consumo de potencia de la red. Permite a todos los dispositivos saber cuándo pueden transmitir. En este modelo, los dos caminos de la red tienen un distribuidor que se encarga de controlar el canal y dirigir las transmisiones. Las balizas que dan nombre a este tipo de entorno, se usan para poder sincronizar todos los dispositivos que conforman la red, identificando la red domótica. Los intervalos de las balizas son asignados por el coordinador de la red y pueden variar desde los 15 ms a los 4 minutos.

  Este modo se recomienda cuándo el coordinador de la red se alimenta con una batería. Los dispositivos de la red escuchan al coordinador durante el lanzamiento de esta baliza, que se podría definir como un mensaje de broadcast, una vez lanzado este mensaje, todo aquel dispositivo que quiera intervenir, lo primero que tendrá que hacer es registrarse para el coordinador, y, es entonces cuando mira si hay mensajes para él. En caso que no haya mensajes, el dispositivo vuelve al modo "dormido", y se despierta de acuerdo a un horario establecido previamente por el coordinador. En cuanto el coordinador termina de lanzar el mensaje baliza, vuelve al modo "dormido".

* **Sin Balizas**

  En este tipo de sistema, cada dispositivo es autónomo, pudiendo iniciar una conversación, en la cual otros nodos pueden intervenir. A veces, puede ocurrir que el dispositivo destino puede no oír la petición, que el canal por el cual se envía el mensaje esté ocupado.

  Este sistema se usa en los sistemas de seguridad, en los cuales, sus sensores y detectores, están en estado "dormido"la mayoría del tiempo. Para saber que todavía están en funcionamiento y no hay ningún problema, los nodos pasan a estado "despierto"de forma regular para anunciar que siguen en la red. Cuando se produce un evento, es decir, cuando alguno de los sensores o detectores, lanza una señal, se transmite la señal de alarma correspondiente. En ese momento, el coordinador de red recibe el mensaje enviado por el sensor, y activa la alarma correspondiente. En este tipo de sistema, el coordinador se alimenta de la red principal durante todo el tiempo.

1.5.3. Protocolo X10

Se trata de un protocolo de comunicación para el control remoto de dispositivos eléctricos que utiliza la línea eléctrica (220V o 11V AC) existente, para transmitir señales de control entre equipos de automatización del hogar (domótica, inmótica) en formato digital. Los dispositivos **X10** que se comercializan son solo para uso individual y en entornos domésticos de hasta 250 m^2, dada su limitación de ancho de banda y el número máximo de equipos que se pueden controlar (256). Aunque, existen nuevos elementos que incorporan, entre otros, los protocolos X-10 extendidos, para dar funcionalidad a soluciones de comunicación como la bidireccional, la solicitud de estados y la comprobación de la correcta transmisión de las tramas.

X10 fue la primera tecnología domótica en aparecer y sigue siendo una de las más importantes hoy en día, principalmente, por la facilidad de su instalación, ya que, no necesita cableado adicional además de la red eléctrica.

Las señales de control del protocolo **X10** se basan en la transmisión de ráfagas de pulsos RF (120 KHz) que representan información digital.

El protocolo **X10** consta de bits de «direcciones» y de «órdenes», por ejemplo, permitiría enviar algo como «lámpara #1», «¡enciéndete!» y el sistema procedería a ejecutar dicha orden. Se puede enviar la misma orden a diferentes direcciones, algunas de las órdenes más importantes son: ON, OFF, All Lights ON, All OFF, DIM o BRIGHT.

Los dispositivos, normalmente están conectados a la red mediante módulos **X10** (receptores). **X10** distingue entre **módulos de lámparas** y **módulos de dispositivos**. Los módulos de lámpara suministran energía y aceptan órdenes X-10, mientras que, los módulos de dispositivos, son capaces de controlar cargas más pesadas, como, por ejemplo, motores o calentadores, simplemente, encendiéndolos o apagándolos.

Un ejemplo de uso de esta tecnología sería, por ejemplo, conectar las luces de una vivienda mediante módulos de lámpara **X10**, y asignar a cada módulo una dirección, por ejemplo L1, así, al enviar la orden «L1 ON» a través de la instalación eléctrica, la luz se debería encender. Uno de los problemas más grandes con esta tecnología, es, el hecho de que, se ve afectada por el ruido eléctrico de la red.

El hardware compatible con este protocolo se divide en 3 grupos.

* **Módulos de Dispositivos**

  Como se ha visto anteriormente, dependiendo de la carga que vaya a tener que soportar, podemos elegir entre, módulos de lámparas o módulos de dispositivos. Los módulos de lámpara, son capaces de regular la carga de la lámpara que controlan, permitiendo así, regular el brillo de la misma. Por su parte, los módulos de dispositivos, cambian la señal mediante un impulso de relé, es decir, solo se pueden controlar de forma total, o encendido o apagado.

  Tambiém existen los módulos de sensores, que detectan e informan de la temperatura de la luz, el movimiento, o las apertura y cierres de contacto.

* **Controladores**

  Los controladores **X10**, pueden ser muy simples o muy complejos, estando, los simples, preparados para controlar cuatro dispositivos **X10** en cuatro direcciones secuenciales (1-4 ó 5-8), a su vez, los más complejos són capaces de controlar un mayor número de unidades y/o incorporar temporizadores que se encargan de realizar funciones preprogramadas en cada momento del día.

* **Puentes**

  Existen puentes para traducir **X10** a otros estándares de domótica, como por ejemplo, **KNX**.

1.5.4. Protocolo Z-Wave

Se trata de un protocolo de comunicación inalámbrico para la domótica moderna, es uno de los protocolos más de moda hoy en día en el mundo de la domótica. Se trata de un protocolo aún sin estandarizar, aunque, existe la alianza **Z-Wave** para asegurar la interoperabilidad de los dispositivos. Se trata de un protocolo cerrado, por tanto, es necesario se miembro de dicha alianza para acceder al mismo, sin embargo, es fácil encontrar documentación de este protocolo.

Este protocolo trabaja en la banda de los 868MHz, evitando así la saturada banda de 2,4GHz. Puede llegar a una velocidad de 40 Kbit/s, llegando a un rango de unos 30m en condiciones ideales. Funciona con una topología de red en malla, tal como se ah visto con el protocolo **ZigBee**, es decir, cada nodo puede ser emisor o receptor.

Uno de los inconvenientes de este tipo de protocolo es el alto consumo eléctrico de los nodos, ya que, el hardware encargado de la comunicación ha de estar activo todo el tiempo.

El sistema define dos tipos de dispositivos:

- **Controladores:** Son aquellos que inician y envían los comandos de control necesarios a los nodos. Los controladores conocen la organización de toda la red **Z-Wave**, pudiéndose comunicar con cualquier conectado a la red. El primer controlador que se instale tomará el papel de controlador primario y será el encargado de crear la red. Tan solo puede existir un controlador por cada red **Z-Wave**, y, solo el controlador tendrá el permiso de añadir o eliminar nodos de la red.

  Existen dos tipos de controladores, los controladores de instalador y los controladores puente. Los primeros son una herramienta utilizada por el instalador para llevar a cabo tareas de configuración y mantenimiento de la red. Y, los segundos, se encargan de la conexión de la red **Z-Wave** a otra redes existentes como, las redes **KNX** o **X10**.

- **Esclavos:** Son los que obedecen, ejecutan y contestan los comandos de los controladores. Un esclavo no puede intercambiar información con otro esclavo directamente sin pasar por el controlador.

Este protocolo aporta grandes ventajas a la domótica, siendo su principal aportación, la de un sistema domótico sin cables para comunicación, aunque, como todo sistema también tiene desventajas, como lo són, en este caso, los requisitos de consumo energético y el mantenimiento. Aunque, muchas empresas están dando un impulso a este protocolo que puede llegar a ser un estándar en la domótica inalámbrica.

1.5.5. Protocolo EnOcean

Por otro lado, existe un protocolo de comunicación inalámbrico, llamado **EnOcean**, que, en contrapartida a **Z-Wave**, se basa en el poco consumo energético de sus componentes, llegando incluso a no hacer falta fuentes de alimentación para sus sensores o actuadores.

Se trata de una tecnología patentada por una empresa alemana, aunque, en estos momentos no se considera un estándar. Sin embargo, los fabricantes de productos domóticos pueden crear dispositivos compatibles con esta tecnología, siempre que la Alianza **EnOcean** les conceda dicha licencia.

Para poder ofrecer dispositivos sin baterías ni fuentes de alimentación, la tecnología **EnOcean** consiste en transformar energía que se encuentre en el entorno del dispositivo en energía eléctrica, es decir, por ejemplo, al pulsar un interruptor domótico **EnOcean**, la energía mecánica que se produce con esa pulsación se convierte en energía eléctrica para alimentar el dispositivo, también se pueden alimentar mediante células fotovoltaicas o otro tipo de energías renovables. Aunque, la mayoría de dispositivos no funcionan con pilas, se pueden agregar para dispositivos en condiciones extremas.

Con esta tecnología, las comunicaciones fluyen por la banda de los 868MHz, al igual que la tecnología **Z-Wave**, teniendo un alcance teórico de 300 metros. A diferencia de **Z-Wave**, los dispositivos **EnOcean**, tan solo reportan su estado al controlador cuando reciben la energía suficiente, en la mayoría de casos, esto se produce al ser accionados. Esta tecnología se puede implementar en sistemas **Z-Wave** o otros tipos de sistemas.

Por tanto, la ventajas de este sistema son el casi nulo consumo energético, así como, el poco mantenimiento que necesitan, así como, el bajo precio de sus productos.

2 Ejemplos Domótica Comercial

Definiríamos como domótica, todo aquel sistema, capaz de recoger información de diferentes sensores y utilizar la misma para enviar órdenes a diferentes actuadores, siendo el sistema capaz de acceder a redes de información. Las diferentes ventajas o contribuciones de la domótica se ven reflejadas en diferentes campos, como son, el ahorro energético, la accesibilidad, la seguridad, el confort y calidad de vida y las comunicaciones.

Definida la palabra domótica, esta está experimentando un "boom"bastante grande en los últimos años, debido, no tanto al interés particular de los consumidores, si no, a la gran cantidad de inversión por parte de grandes empresas como **Samsung**, **Apple** o **Google**, ya que, ven en este un nuevo mercado en el que tener parte. Este crecimiento también es debido a que, cada vez más, vivimos en una sociedad conectada, dónde a todas horas tenemos una conexión a internet disponible, ya sea, desde un smartphone, un PC o una televisión, por tanto, el próximo paso lógico en esto, es el de conectar a la red todos los demás dispositivos de los que disponemos en nuestro entorno, es decir, todo tipo de electrodomésticos, diferentes motores de los que disponemos en una vivienda (Motores de persianas, Motores de puertas de garaje, etc..), así como, todo tipo de iluminación, fontanería o sistema de climatización o calefacción.

Otra de las razones de este crecimiento, ha sido, la disminución del precio medio de una instalación domótica, ya que, hace pocos años, el precio de este tipo de tecnología era muy elevado, viéndose este reducido en los últimos años. A la vez que ha bajado el precio, también han aumentado las funcionalidades que se pueden obtener de un sistema domótico, por ejemplo, la capacidad de conocer el consumo real de nuestros aparatos, conocido como, Smart Metering.

Hoy en día, se pueden encontrar en el mercado, infinidad de productos o sistemas domóticos, ya sean, productos de grandes empresas, sistemas operativos de software abierto o propietario, aplicaciones móviles enfocadas a la domótica o todo tipo de sistemas "Do It Yourself"para que sea el consumidor quién monte su propio sistema domótico, dónde podríamos incluir todo tipo de variantes de sistemas **Arduino** o **Raspberry Pi**.

2.1. Apple Homekit

Esta es la propuesta para domótica de **Apple**, este sistema, a diferencia de la mayoría, no cuenta ni necesita una centralita o nodo central que se encargue del control de todos los dispositivos conectados a la red domótica, ya que, encarga esta tarea a cualquier dispositivo iOS del que se disponga, ya sea, un iPad o un iPhone. Estos es posible gracias al uso de iCloud, es decir, el servicio de cloud computing de **Apple**.

Los diferentes productos que se obtengan para la red domótica tienen que ser compatibles con la tecnología **Homekit** para poder funcionar, es decir, para que un producto de una empresa sea compatible con esta tecnología, dicha empresa, tendría que pagar unos derechos de licencia a **Apple**, para permitir que sus productos sean compatibles con **Homekit**.

Existen infinidad de productos de diferentes compañías compatibles con **Homekit**, algunos ejemplos serían los siguientes.

(a) Cámara IP

(b) Enchufe Inteligente

(c) Philips Hue

(d) Termostato Inteligente

(e) Sensor de Movimiento

Figura 1: Productos Compatibles Homekit

2.2. Samsung Smart Home

Otra de las propuestas de las grandes empresas tecnológicas, en concreto, **Samsung**, tal como se ha visto con **Apple**, el sistema de **Samsung** también esta basado en la nube, permitiendo controlar todos los productos domóticos de nuestro hogar, desde cualquier dispositivo, aunque, en este caso, desde cualquier dispositivo Android, no es necesario que sea un producto **Samsung**, debido a que, existe una aplicación gratuita en Play Store, llamada **Samsung** Smart Home, que permite controlar todos nuestros aparatos domóticos.

Para conseguir este objetivo, **Samsung** creó un protocolo de software, el Smart Home Software Protocol, que permite, que, todas aquellas empresas que quieran crear productos domóticos compatibles con el sistema de **Samsung**, usen ese protocolo de software. A su vez, la mayoría de productos domóticos de **Samsung**, se pueden controlar con tecnologías de otras empresas, tales como, **Google** Home de **Google**, o Alexa de **Amazon**.

Hace pocos años, **Samsung** adquirió la empresa SmartThings, dando así un impulso a su inversión en tecnologías domóticas, y, cambiando un poco el modo de uso de esta tecnología. En la actualidad, el sistema domótico de **Samsung**, se centra en el uso de un nodo central, del cual existen varias opciones, ya sea el nodo fabricado por SmartTings, por **Samsung** o por **Google** o **Amazon**.

Ilustro ahora, alguno de los productos domóticos de **Samsung**, o de fabricantes asociados.

(a) Video Portero Inteligente

(b) Sensor de Apertura de Puertas

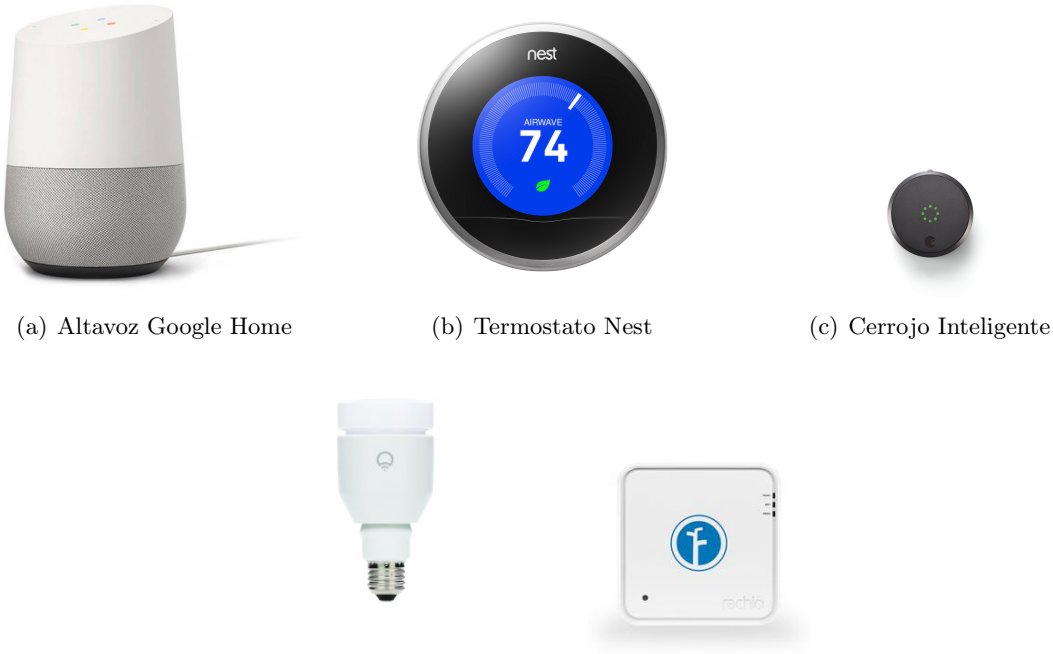
(c) Hub SmartThings

Figura 2: Productos Compatibles Samsung Smart Home

2.3. Google Home

El sistema domótico de **Google**, se controla mediante un altavoz inteligente llamado **Google** Home, el cual, integra el asistente de **Google**, **Google** Assistant. Dicho altavoz, es capaz de reconocer diferentes voces de los diferentes usuarios o inquilinos de una vivienda, y, realizar diferentes acciones mediante comandos de voz naturales, es decir, por ejemplo, utilizando un Chromecast, es posible, con un simple comando de voz, reproducir diferente contenido en nuestra televisión, o controlar la temperatura de nuestra vivienda con el termostato de la empresa propiedad de **Google**, Nest. Este sistema, también es compatible con la mayoría de productos compatibles con **Samsung** Smart Home.

Algunos de los productos compatibles con **Google** Home.



(a) Altavoz Google Home (b) Termostato Nest (c) Cerrojo Inteligente

(d) Iluminación LIFX (e) Controlador Aspersores

Figura 3: Productos Compatibles Google Home

2.4. Amazon Echo

Amazon Echo, es el sistema domótico de **Amazon**, y, es muy parecido al enfoque de **Google**, contando también, con su propio altavoz inteligente unido a un asistente de voz, en este caso, Alexa, el cual, se encarga de obedecer las órdenes del usuario, el cual se comunica mediante comandos de voz para controlar los dispositivos domóticos de la vivienda.

En este caso, son casi los mismos los dispositivos compatibles con **Amazon** Echo, que los compatibles con **Samsung** o con **Google**, ya que, como no existe una tecnología que domine el mercado con claridad, los fabricantes de dispostivos domóticos, crean sus productos para ser compatibles con la mayoría de los sistemas de control domótico que existen en el mercado.

Algunas fotografías de dispositivos compatibles con **Amazon** Echo.

(a) Amazon Echo y Echo Dot (b) Interruptor WeMo (c) Iluminación Philips Hue

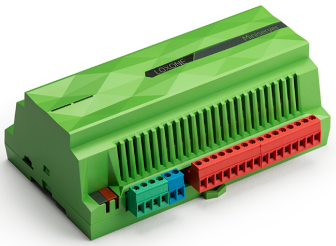
Figura 4: Productos Compatibles Amazon Echo

2.5. Loxone

Loxone es una empresa Austríaca, que, ha desarrollado un sistema domótico muy amplio y robusto. Se basa en un miniserver con un número determinado de entradas y salidas digitales y analógicas, el cual, se encarga del control del sistema domótico, ya que, todos los dispositivos de la red domótica se han de conectar a este miniserver.

Existen un gran número de extensiones para este miniserver, ya sean, para ampliar el número de entradas o salidas, para conectar el sistema a otros sistemas o protocolos domóticos, como **KNX** o Modbus, o, para dotar al sistema de comunicación inalámbrica.

Este sistema dispone de diferentes dispositivos compatibles con el mismo, desde, iluminación a pulsadores táctiles, pasando por, válvulas o cables de carga para cargar coches eléctricos, o, diferentes tipos de sensores, de presencia, detección de humos, de temperatura o detectores de rotura de cristales.



(a) Miniserver Loxone

(b) Pulsador Loxone

(c) Sensor de Temperatura Loxone

Figura 5: Productos Domóticos Loxone

3 Ejemplos Domótica Open Hardware

Definiríamos como domótica **Open-Hardware**, todo aquel sistema domótico que hace uso de tecnologías o hardware abierto, concepto que se ha definido al inicio de esta memoria. En este conjunto de sistemas, podemos encontrar infinidad de ejemplos de domótica realizados con placas **Arduino** o **Raspberry Pi**, y todas sus variantes. Este tipo de sistemas no suelen ir dirigidos al gran público, si no que, van dirigidos más a personas con ciertos conocimientos técnicos.

Un ejemplo de esto, es todo el movimiento "Maker", que, cada vez más, vemos en nuestra sociedad, este movimiento se basa en la filosofía **DIY**, es decir, la filosofía que anima al consumidor a fabricar o modificar todo tipo de cosas por él mismo, permitiendo el ahorro de dinero, el entretenimiento y el aprendizaje. Más profundamente, este movimiento se basa en una lógica anticapitalista, ya que, promueve la creación, reutilización o la reparación de todo tipo de objetos, dejando así, de lado, la lógica capitalista y consumista del comprar-tirar-comprar.

En concreto, en este proyecto, se hace uso de la filosofía **Open-Hardware** y del **DIY**, ya que, todos los materiales que se utilizan se basan en estos conceptos, pero, antes de entrar más en profundidad en este proyecto, paso a detallar algunos de los sistemas más conocidos y utilizados dentro de la domótica **Open-Hardware**.

3.1. Arduino

El sistema más conocido y utilizado en la domótica **Open-Hardware**, y en la mayoría de proyectos de electrónica, es, sin duda, **Arduino**. Este sistema se basa en el uso de una placa de desarrollo que incorpora un microcontrolador, así como, diferentes puertos digitales y analógicos de entrada salida de propósito general o **GPIO**, así como, el uso de un IDE de código abierto para poder realizar los programas con los que controlar **Arduino** y los diferentes elementos que se pueden conectar a la placa.

Todos los diferentes tipos de **Arduino** que existen, están basados en microcontroladores Atmel y se programan un lenguaje basado en el lenguaje Wiring, que a su vez, está basado en lenguaje C. Para poder cargar los programas en la placa, esta dispone de diferentes tipos de puertos USB.

Existen diferentes tipos de placas **Arduino**, entre los más conocidos y utilizados se encuentran.

- **Arduino Uno:** Esta es la variante más conocida y utilizada de **Arduino**, dispone de 20 pines **GPIO**, de los cuales, 6 pines son de entrada o salida analógicos. Se alimenta con una fuente de 5V y dispone de un microcontrolador ATmega328 a una velocidad de 16MHz.

- **Arduino Mega:** Es una placa **Arduino**, muy parecida a la **Arduino** Uno, pero más grande, ya que, cuenta con 70 pines **GPIO**, 16 de ellos analógicos. Se alimenta en un rango de 5 a 12V, y hace uso del microcontrolador ATmega2560 a 16MHz.

- **Arduino Due:** Se trata de la primera placa **Arduino** que dispone de un microcontrolador de 32-bit, funciona a 3.3V y dispone de 66 pines **GPIO**, 12 de ellos analógicos, funciona gracias al micrcontrolador AT91SAM3X8E, el cual, se basa en una arquitectura ARM Cortex-M3 y funciona a 84MHz.

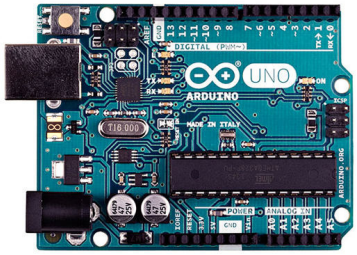
- **Arduino Micro:** Es una versión miniaturizada de **Arduino**, la cual dispone de 20 pines **GPIO**, 12 de ellos analógicos, conexión Micro-Usb como sus versiones más grandes, así como, un voltaje de entrada de 7 a 12V. Se basa en el microcontrolador ATmega32u4, el cual funciona a 16MHz.

- **Arduino Nano:** Otra de las placas más pequeñas de la familia **Arduino**, dispone de 22 pines **GPIO**, 8 de ellos analógicos, se basa en el microcontrolador ATmega328, al igual que el **Arduino** Uno, el cual funciona a una velocidad de 16MHz.

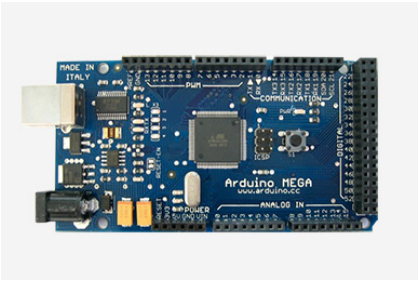
- **Arduino Mini:** Es la placa **Arduino** más pequeña que existe en la actualidad, hace uso del mismo microcontrolador ATmega328 a 16MHz, pero, la mayor diferencia con el resto de la familia, es el hecho de que no dispone de una conexión USB en la placa, y se tiene que programar mediante el uso de un adaptador USb a Serial, conectado este a los pines de comunicación de los que dispone la placa. El número de pines **GPIO** de esta placa es de 22, 8 analógicos, y su voltaje de entrada máximo son 9V.

- **Arduino 101:** Una de las últimas placas presentadas por la empresa que diseña **Arduino**, es el **Arduino** 101, el cual, solo está disponible con esa nomenclatura en los Estados Unidos, fuera de allí se conoce como Genuino 101. Esta placa dispone de diferentes mejoras respecto a los demás **Arduino**, como pueden ser, la inclusión de conectividad Bluetooth Low-Energy y un acelerómetro de 6 ejes, además dispone de dos núcleos en la misma placa, uno con arquitectura x86 y uno con una arquitectura de 32-bit, los dos con una velocidad de reloj de 32MHz. La placa dispone de 20 pines **GPIO**, 6 de ellos analógicos y se alimenta con 3.3V. Esta placa ha sido diseñada con la colaboración de Intel.

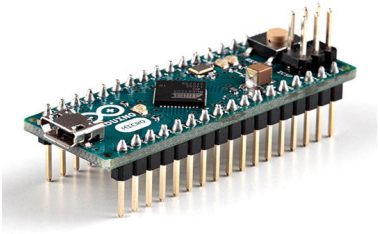
Algunas fotografías de los diferentes tipos de placas **Arduino** que he descrito.



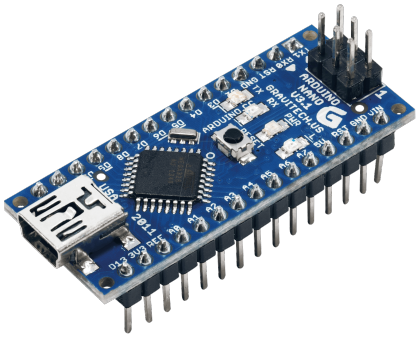
(a) Arduino Uno



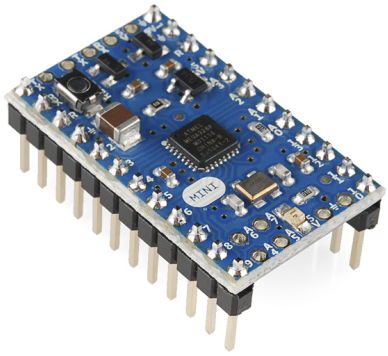
(b) Arduino Mega



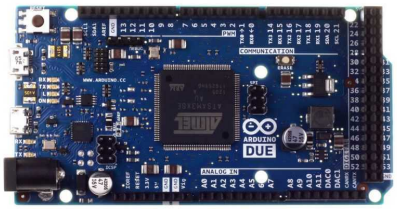
(c) Arduino Micro



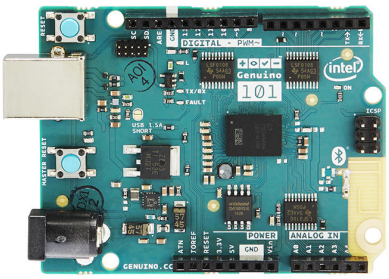
(d) Arduino Nano



(e) Arduino Mini



(f) Arduino Due



(g) Arduino 101

Figura 6: Tipos diferentes de Arduino

La mayoría de proyectos de domótica basados en **Arduino**, se tratan de proyectos caseros y personalizados para las necesidades concretas de la persona que crea y diseña el sistema, aunque, existen algunos proyectos interesantes que intentan llevar la domótica basada en **Arduino** a todo el mundo.

3.1.1. Souliss

Uno de las soluciones domóticas basadas en **Arduino** más maduras y robustas es, **Souliss**, la cual se basa en la integración de diferentes nodos basados en **Arduino**, estos nodos están controlados por un nodo central, una configuración muy parecida a una red ZigBee, la cual se ha detallado anteriormente, siendo diferencia principal el precio de los nodos, mucho más baratos con este tipo de tecnología que con ZigBee. Con Souliss se pueden controlar todo tipo de productos y aparatos de los que se dispone en un hogar típico, tales como, luces, relés, motores, dispositivos controlados por infrarrojos o alarmas, desde cualquier dispositivo móvil, ya sea un Smartphone o una Tablet.

3.2. Raspberry Pi

Raspberry Pi es más un computador en miniatura que una placa microcontroladora, pero, debido a la gran comunidad de gente que tiene y usa la **Raspberry Pi**, además del hecho de que dispone de 40 pines de entrada salida de propósito general, es posible, utilizar-la como microcontroladora para un sistema domótico.

Raspberry Pi no es hardware libre propiamente dicho, ya que, la fundación **Raspberry Pi** es quién tiene la propiedad del diseño registrada, pero, permite el uso libre de esta placa tanto a nivel educativo como a nivel particular o empresarial. Dicho esto, si es un sistema basado en software libre, ya que, utiliza un sistema operativo basado en Debian, Raspbian, aunque, permite ser gobernado por diferentes sistemas operativos, entre ellos, una versión reducida de Windows 10.

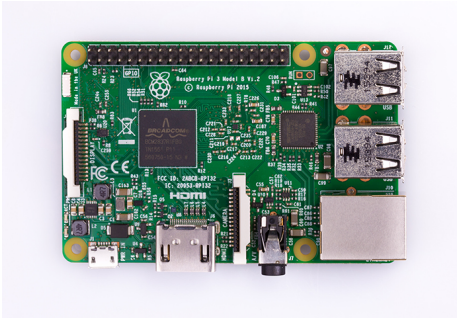
En términos de potencia, **Raspberry Pi**, es mucho más potente que **Arduino**, debido a que, utiliza un SoC fabricado por Broadcom, en concreto, la última versión de **Raspberry Pi**, la **Raspberry Pi** 3, utiliza un SoC BCM2837, con 4 núcleos a una velocidad de reloj de 1.2GHz, dicho SoC, este SoC se basa en una arquitectura ARM de 64 Bits. Además la **Raspberry Pi** 3 dispone de 1GB de memoria RAM, conectividad Wi-Fi y Bluetooth integrada, así como, varios puertos USB, una salida de vídeo HDMI, conector jack de audio de 3.5mm y puerto Ethernet.

Al igual que ocurre con **Arduino**, existen en el mercado diferentes modelos del Micro-PC **Raspberry Pi**, siendo los más destacados los siguientes.

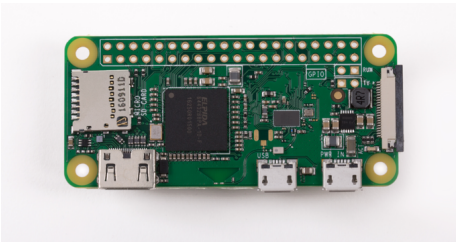
- **Raspberry Pi 3 Model B:** Esta es la tercera generación de la **Raspberry Pi**, y, por ahora, la última novedad, como se ha descrito anteriormente, dispone de una CPU de 4 núcleos, conectividad Wi-FI, Bluetooth y Ethernet, 4 puertos USB, puerto HDMI y 1 GB de memoria RAM.

- **Raspberry Pi Zero W:** Se trata de una versión en miniatura de la **Raspberry Pi**, contando con una CPU Single-Core a una velocidad de reloj de 1GHz, además de, conectividad Wi-Fi, Bluetooth, 512 MB de RAM, un puerto mini-HDMI, un puerto micro-USB para conectar periféricos y otro puerto micro-USB para alimentación. Al igual que la **Raspberry Pi** de tamaño completo, esta cuenta con los mismos pines **GPIO**.
- **Raspberry Pi Compute Module:** Se trata de un producto enfocado para la industria, ya que, se trata de una **Raspberry Pi** 3 en formato de conexión SO-DIMM, y, se usa básicamente para industrias o empresas que quieren diseñar su propio PCB y integrar en él este módulo.

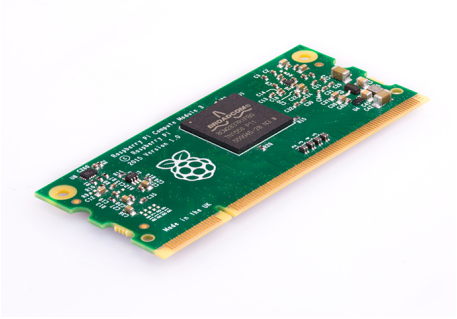
Los diferentes modelos y placas de la fundación **Raspberry Pi**.



(a) Raspberry Pi 3 Model B



(b) Raspberry Pi Zero W



(c) Raspberry Pi 3 Compute Module

Figura 7: Tipos diferentes de Placas Raspberry Pi

En el caso de la domótica con **Raspberry Pi**, los proyectos no se centran tanto en el hardware, si no, en el software, ya que, el uso perfecto de la **Raspberry Pi**, es, como cerebro de un sistema domótico, así pues, se pueden encontrar varios proyectos o empresas que se dedican a desarrollar o mantener sistemas domóticos controlados con Micro-PC's **Raspberry Pi**. Algunos de de estos proyectos son.

3.2.1. DomoticZ

Se trata de un sistema disponible para Windows, Linux, además de, para dispositivos embebidos como **Raspberry Pi**, ya que, utiliza un servidor web HTML5. Este software es compatible con muchos de los protocolos disponibles en el mercado, tales como, Z-Wave, X10, EnOcean, etc...

Dispone de la opción de programar scripts con el lenguaje Lua, y tiene una interfaz de usuario bastante sencilla, la cual, se puede controlar desde dispositivos móviles, tanto iOS como Android.

Aunque, para poder controlar el sistema desde un Smartphone, se ha de contar con un dispositivo que haga de interfaz entre los módulos del sistema y el protocolo dómotico que se elija, en este caso, ese dispositivo sería la **Raspberry Pi**, ya que, es un dispositivo compatible, siempre y cuando, se utilice una tarjeta que se conecta a los puertos **GPIO** de la misma, llamada RaZberry.

3.2.2. PIHome

Otro sistema basado en **Raspberry Pi**, es PIHome, aunque, este, está en una fase un poco más temprana que DomoticZ, ya que, por ahora, tan solo permite el control de enchufes inteligentes para controlar el sistema domótico.

Se trata de una solución basada en web, la cual, funciona en **Raspberry Pi** y permite el control del sistema gracias a un transmisor inalámbrico. Este software está programado en lenguaje HTML, CSS; JavaScript y PHP, en una nueva versión permite el programado del encendido o apagado de luces gracias al uso de la API de Google Calendar.

3.2.3. JEEDOM

Este es uno de los sistemas domóticos **Open-Hardware** más utilizados y con mayor catálogo de productos compatibles del mercado, se trata de una solución basada en **Raspberry Pi**, compatible con el protocolo Z-Wave y EnOcean. Este sistema permite la creación a medida de cada solución domótica.

Además, dispone de aplicaciones móviles para iOS y Android que permiten el control del sistema domótico. Este tipo de sistema, al ser de código abierto, permite su instalación en una **Raspberry Pi** que se tenga disponible, aunque, además, es posible comprar una con todo instalado y comprobado, totalmente plug&play.

4 Materiales Utilizados En El Proyecto

4.1. Tipos de Microcontroladores Investigados

Antes de empezar a realizar el proyecto, lo primero que hice fue el trabajo de investigación necesario para poder elegir una placa de desarrollo que se adaptara a mis necesidades y las del proyecto. Al ser un mercado bastante grande debido a la gran cantidad de placas existentes, véase, **Arduino**, **Raspberry Pi**, **NodeMCU**, **Nanode**, **Teensy**, etc...

- **Arduino Uno:** La placa de desarrollo por excelencia, es la más utilizada, debido a su reducido precio, la facilidad de uso y programación con su IDE oficial, además, gracias a estas razones, tiene una gran comunidad de usuarios y gran cantidad de foros en los que se resuelve casi cualquier duda. Los programas para poder utilizar los pines GPIO de entrada/salida, se programan con el IDE **Arduino**, en lenguaje C++.

  Arduino Uno, no tiene conectividad por si misma, pero, se pueden comprar módulos por separado, para dotar a la placa de conectividad Ethernet, Wi-Fi, Radiofrecuencia, ampliar los pines de entrada/salida, etc...

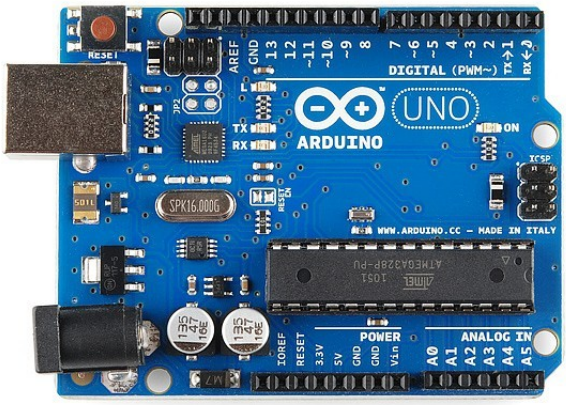


Figura 8: Arduino Uno

Microcontrolador/CPU	ATmega328P
Voltaje de Entrada	7-12V
Voltaje De Entrada Límite	6-20V
Voltaje Operativo	5V
Pines Digitales Entrada/Salida	14
Pines Digitales PWM de Entrada/Salida	6
Pines Entrada Analógicos	6
Corriente Directa Pines Entrada/Salida	20 mA
Corriente Directa Pines 3.3V	50 mA
Memoria Flash	32 KB
SRAM	2 KB
EEPROM	1 KB
Velocidad de Reloj CPU	16 MHz
Conectividad	Módulos
Longitud	68.6 mm
Anchura	53.4 mm
Peso	25 g

Cuadro 1: Especificaciones Técnicas Arduino Uno

- **Raspberry Pi:** Otra de las opciones más utilizadas en el mundo de las placas de desarrollo, aunque, esta se encuadra más en la categoría de Micro-PC's que en la de microcontroladores, ya que, es mucho más potente que cualquier microcontrolador, no obstante, no deja de ser una opción en este determinado proyecto por el hecho de contar con pines de entrada/salida de propósito general, conocidos cómo GPIO. Este tipo de placas también tienen una gran comunidad de usuarios y foros dónde encontrar ayuda y soluciones a diferentes problemas. Se basan en una arquitectura ARM, y se controlan mediante diferentes distribuciones de Linux, tales como, Raspbian o Snappy Ubuntu. Los programas para poder controlar sus pines GPIO se pueden programar en diferentes lenguajes, como, Python o C++.

  Existen diferentes modelos y tamaños en los que existe este Micro-PC, siendo el más reciente, la **Raspberry Pi** Model 3.

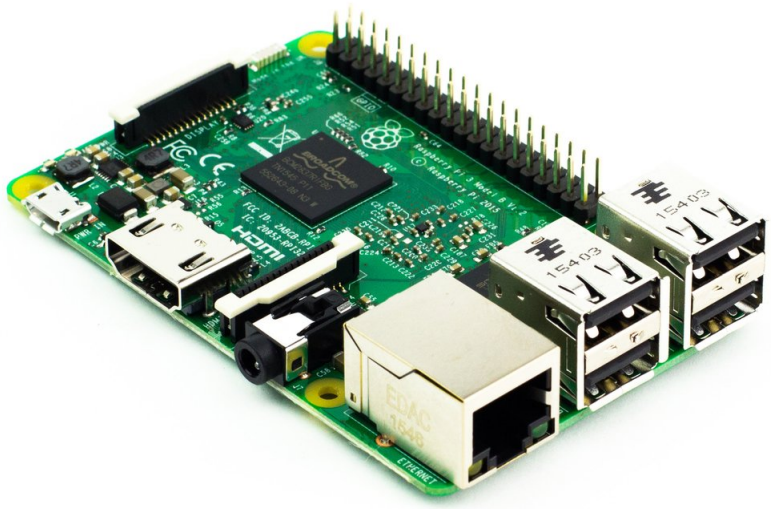


Figura 9: Raspberry Pi Model 3

Microcontrolador/CPU	Broadcom BCM2837
Voltaje de Entrada	5V
Pines Digitales Entrada/Salida	40
Corriente Directa Pines Entrada/Salida	16 mA
Memoria RAM/VRAM	1 GB
Velocidad de Reloj CPU	4 x 1.2GHz
Conectividad	Ethernet, Wi-Fi, Bluetooth, USB, HDMI, Jack 3.5 mm
Longitud	85 mm
Anchura	53 mm
Peso	42 g

Cuadro 2: Especificaciones Técnicas Raspberry Pi Model 3

- **Nanode:** Esta es otra de las opciones dentro del mundo de los microcontroladores, se trata de la creación de un Ingeniero Eléctronico de Reino Unido, Ken Boak, el cuál diseñó este microcontrolador,basándose en el mismo chipset que usa **Arduino**. Este producto, tiene como principal diferencia con **Arduino**, la inclusión de un puerto Ethernet integrado en la misma placa, por tanto, se ahorra el uso de un Shield Ethernet, además de venir completamente desmontado, por lo tanto, antes de poder usarlo se han de soldar todos sus componentes en la placa base que viene incluida. La mayoría de las demás características se parecen mucho a las del **Arduino** Uno, debido al uso del mismo chipset.

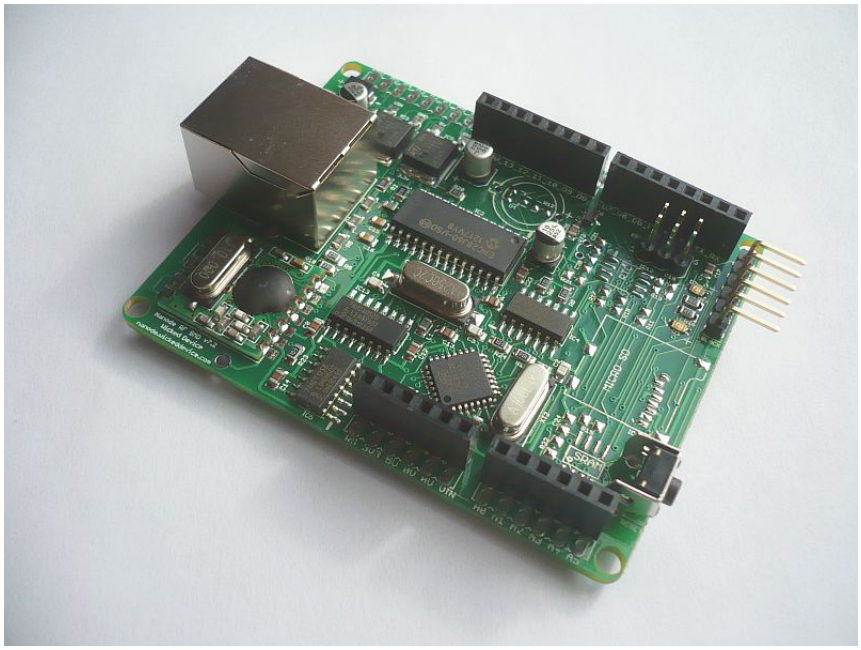


Figura 10: Nanode Gateway

Microcontrolador/CPU	ATmega328P
Voltaje de Entrada	7-12V
Voltaje De Entrada Límite	6-20V
Voltaje Operativo	5V
Pines Digitales Entrada/Salida	28
Pines Digitales PWM de Entrada/Salida	6
Pines Entrada Analógicos	14
Corriente Directa Pines Entrada/Salida	20 mA
Corriente Directa Pines 3.3V	50 mA
Memoria Flash	32 KB
SRAM	2 KB
EEPROM	1 KB
Velocidad de Reloj CPU	16 MHz
Conectividad	ENC28J60 Ethernet Controller
Longitud	63 mm
Anchura	55 mm

Cuadro 3: Especificaciones Técnicas Nanode Gateway

- **Teensy:** Otra opción, una placa poco conocida, pero no por eso peor que **Arduino**, esta placa cuenta con un chipset más potente que el de **Arduino**, aunque, no cuenta con ningún tipo de conectividad más que el puerto Micro-USB para programarla. Se puede programar en C, o con el mismo IDE que **Arduino**, mediante un add-on del mismo. Su principal característica, por tanto, es su reducido tamaño y altas prestaciones.

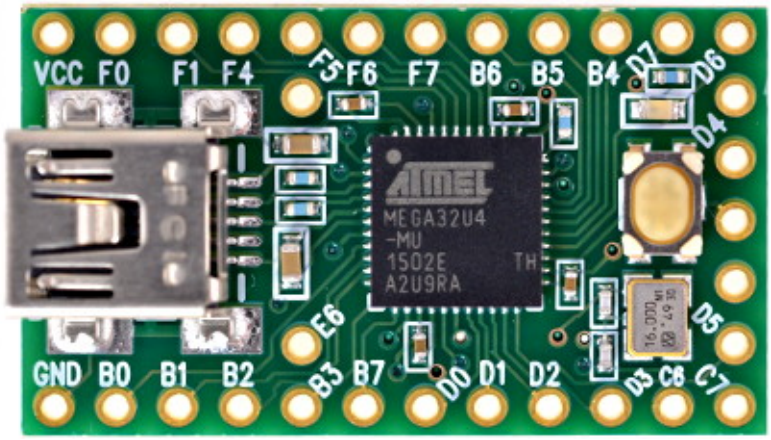


Figura 11: Teensy 2.0

Microcontrolador/CPU	ATMEGA32U4
Voltaje de Entrada	7-12V
Voltaje De Entrada Límite	6-20V
Voltaje Operativo	5V
Pines Digitales Entrada/Salida	25
Pines Digitales PWM de Entrada/Salida	7
Pines Entrada Analógicos	12
Corriente Directa Pines Entrada/Salida	20 mA
Corriente Directa Pines 3.3V	50 mA
Memoria Flash	32 KB
SRAM	2 KB
EEPROM	1 KB
Velocidad de Reloj CPU	16 MHz

Cuadro 4: Especificaciones Técnicas Teensy 2.0

- **NodeMCU:** Finalmente, otra posibilidad fue la placa basada en el SoC Wi-Fi ESP8266, el cuál es fabricado por Espressif. Por tanto, cuenta con conectividad Wi-Fi, suficientes pines GPIO para un proyecto pequeño, un consumo ínfimo y a un precio muy competitivo, hecho este, que le ha valido para crear una comunidad de usuarios bastante grande. Esta SoC, se empezó a utilizar conjuntamente con un **Arduino**, es decir, se usaba este chip para dotar a **Arduino** de conectividad Wi-Fi, pero, al darse cuenta que, este SoC cuenta con una unidad de microcontrolador, apareció esta placa, prescindiendo de **Arduino** para así, en una placa de reducidas dimensiones, contar con un Hardware capaz de conectarse a una red Wi-FI y a su vez funcionar como microcontrolador. En concreto, la placa de la que hablo aquí y la que finalmente he utilizado, es la tercera iteración de esta placa, la **NodeMCU** V3 LoLin, la cual, se basa en el chipset ESP-12, una variante del ESP8266 que cuenta con 11 pines GPIO y 1 convertidor Analógico-Digital, así como, un alcance Wi-Fi de unos 30 metros. Además, gracias al trabajo de la comunidad de usuarios, se puede programar este microcontrolador mediante el IDE de **Arduino**, gracias a un add-on. Por tanto, podríamos decir que se trata de un **Arduino** con conectividad Wi-Fi integrada y a un precio inferior a un **Arduino** Uno.

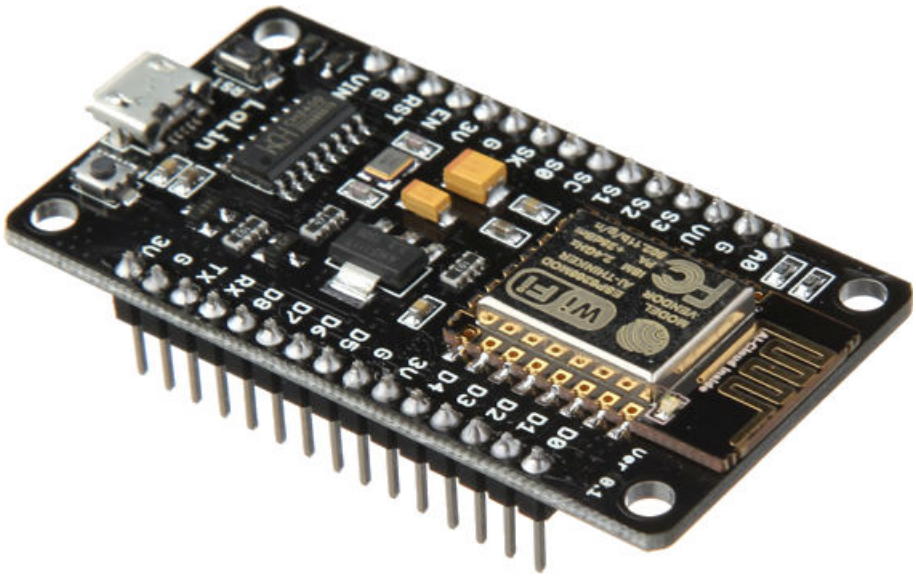


Figura 12: NodeMCU V3 LoLin

Microcontrolador/CPU	Tensilica L106 32 bit
Voltaje de Entrada	3.3V
Voltaje De Entrada Límite	3.3-5V
Pines Digitales Entrada/Salida	17
Pines Entrada Analógicos	1
Memoria Flash	16 MB
SRAM	32 KB
Conectividad	Wi-Fi 802.11 b/g/n, Micro-USB
Velocidad de Reloj CPU	80-160 MHz

Cuadro 5: Especificaciones Técnicas NodeMCU V3 LoLin

4.2. Microcontrolador Utilizado

Una vez vistas las características de los diferentes microcontroladores, decido utilizar la placa de desarrollo de **NodeMCU**, debido, básicamente, a que dispone de conectividad Wi-Fi integrada y tiene un precio bastante bajo, además de, que, se trata de una placa de desarrollo bastante novedosa en el mercado. Aunque, esta placa también tiene sus puntos flacos, como son, la diferencia de pines GPIO con sus competidores y el hecho de que hay pocos proyectos y documentación hechos con esta placa, en comparación a los que hay con **Arduino** por ejemplo.

Esta placa se basa en el SoC ESP-12E fabricado por Espressif Systems, dicho SoC, se encuadra dentro de una familia bastante amplia de módulos fabricados por esta empresa, los cuáles, están todos basados en el mismo chipset, en concreto, el módulo Wi-FI, ESP8266, siendo la principal diferencia entre unos y otros el tamaño y el número de pines que podemos utilizar para propósitos generales.

El **ESP-01** sería la primera iteración de esta familia, tratándose de un módulo creado para dotar de conectividad Wi-Fi a placas como **Arduino** mediante el uso de comandos de comunicación AT que se transmiten a través del puerto serie, este lenguaje de comandos fue desarrollado por la compañía Hayes Communications y se utilizaba para la configuración de módems. La principal pega de este módulo es el número de pines de los que dispone, 8, todos ellos utilizados para la conexión con **Arduino**, es decir, no tiene ningún pin GPIO.

El **ESP-12E** sería la última iteración de esta familia hasta el momento, siendo este módulo, el que se utiliza en la placa **NodeMCU** que voy a utilizar, la principal diferencia con el módulo mencionado anteriormente, es el número de pines GPIO de los que dispone y el factor de forma.

Es decir, existen 12 iteraciones de esta familia de módulos Wi-Fi, muy parecidas entre sí, pero con diferencias en el número de pines GPIO, el tamaño de la memoria RAM y Flash y el factor de forma y conexiones utilizados.

Una de las diferencias con las placas **Arduino**, es la disposición de los pines de entrada salida, lo que se conoce como Pinout. En esta placa, el pinout se ilustra con la imagen siguiente.

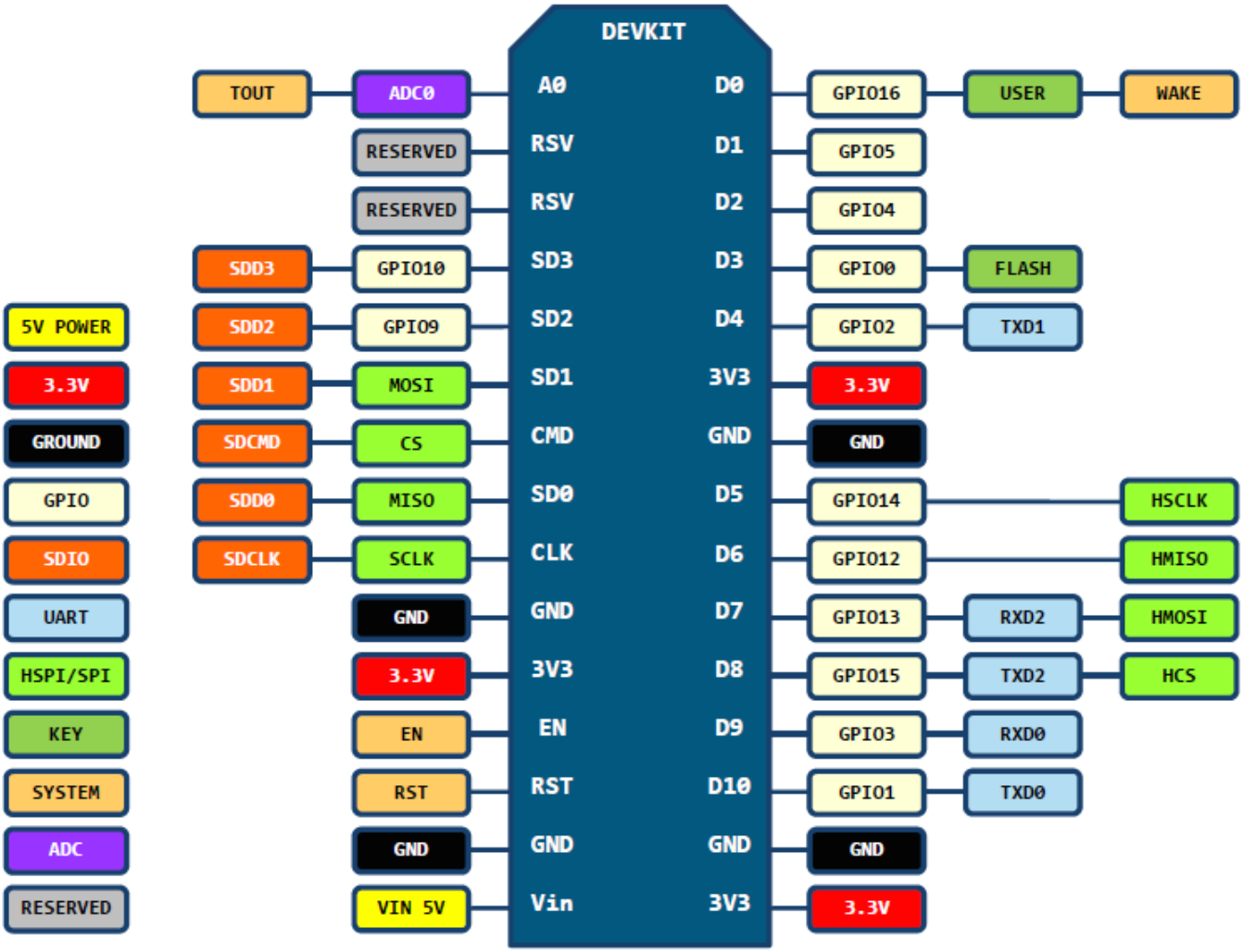


Figura 13: NodeMCU Pinout

4.3. Sensores

Se entiende por sensor, todo aquel objeto capaz de detectar magnitudes, ya sean físicas o químicas, y transformar dichas magnitudes en variables eléctricas. Existen una gran cantidad de sensores, desde sensores de movimiento a sensores de detección de gases, o, sensores magnéticos.

Paso a enumerar los sensores que utilizo para realizar la maqueta domótica de la que tratará este proyecto.

* **Sensor Temperatura:** En concreto, para monitorizar la temperatura en esta maqueta, voy a utilizar el sensor **LM35**, este sensor permite un rango de medición de -55ºC a 150ºC. La medición de temperatura con este sensor se obtiene de la medición de su resistencia eléctrica. Dicho sensor, dispone de su propio circuito de control, el cual, proporciona una salida de voltaje proporcional a la temperatura, en concreto, se incrementa el valor de la salida de voltaje de la siguiente manera, 10mV a cada grado centígrado de diferencia.

  Ilustro este sensor con una fotografía.

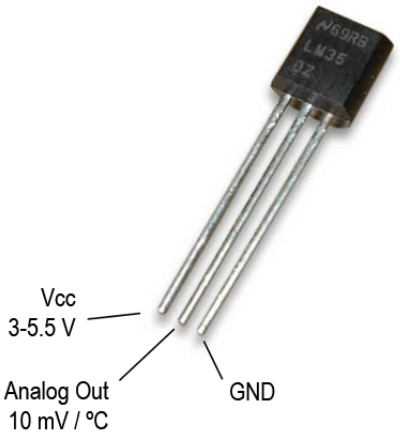


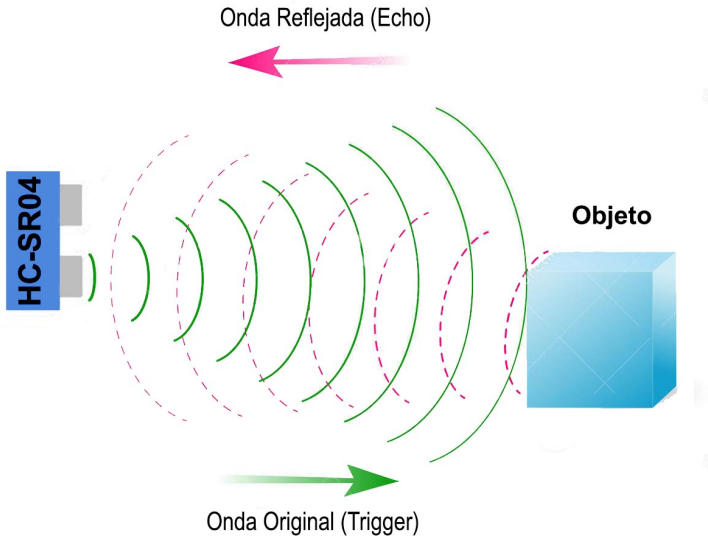
Figura 14: Sensor Temperatura LM35

* **Sensor Distancia:** Se trata de un sensor de medición de distancia, que funciona mediante ultrasonidos. Dispone de un emisor y de un receptor de ultrasonidos en el mismo circuito integrado, y funciona de la siguiente manera. El emisor emite ultrasonido constantemente, y, cuando el receptor recibe ese ultrasonido, se sabe que existe un obstáculo delante del sensor, por tanto, se usa el mismo principio que usan los murciélagos o los sonares de los submarinos.

  En concreto, en este caso, voy a utilizar el sensor de distancia **HC-SR04**, las características de este son, un rango de medición de 2 a 400cm, una frecuencia de ultrasonido de 40KHz, imperceptible para el oído humano y un ángulo de recepción de la señal de 15º, es decir, necesita que el obstáculo que ha de recibir esté, relativamente centrado al sensor.

  Ilustro este sensor y su funcionamiento con unas fotografías.

(a) Sensor Distancia HR-SR04



(b) Funcionamiento Sensor de Distancia

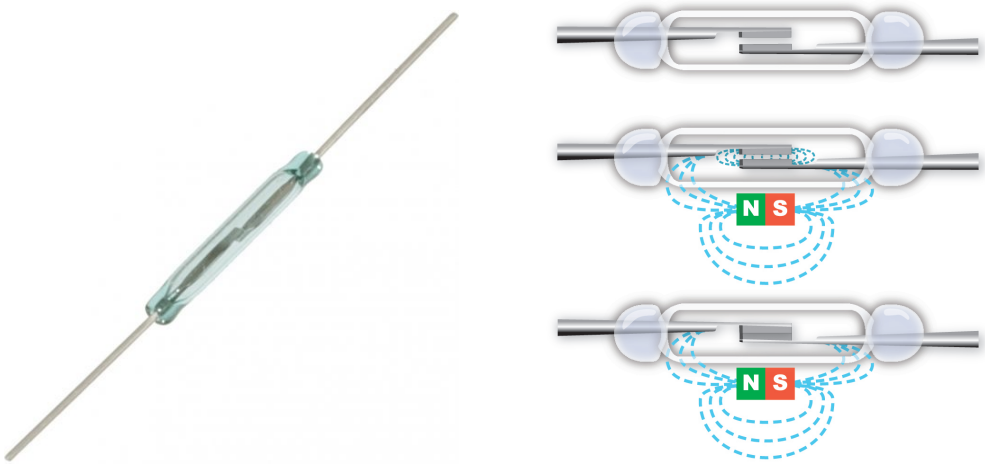
Figura 15: HC-SR04

* **Interruptor Magnético:** Aunque se trata de un interruptor, en este caso, funciona como sensor, ya que, lo voy a utilizar para que envíe una señal al microcontrolador cuándo la puerta principal esté abierta.

  Este dispositivo, se trata de un dispositivo electromecánico que se comporta como un interruptor, el cual, se activa en la presencia de un imán. Como se ha visto en el apartado de la domótica comercial, este tipo de interruptores son ampliamente utilizados para sistemas domóticos, en forma de alarmas, ya que, permiten saber cuando una puerta o una ventana están abiertas o cerradas.

  La principal ventaja de estos interruptores es su bajo precio, y su principal desventaja, el alto tiempo de conmutación que tienen, del orden de 1 a 5 milisegundos, debido, principalmente a ser dispositivos elecromecánicos.

  El principio básico de funcionamiento de este dispositivo es el siguiente, se trata de un interruptor constituido por dos elementos ferromagnéticos de níquel, que se ubican dentro de una ampolla de vidrio sellada, al acercar un campo magnético, provocado, por ejemplo por un imán, la fuerza generada por este campo, provoca que los dos elementos ferromagnéticos entren en contacto, provocando el cierre del circuito eléctrico y dejando pasar la corriente. De esta manera, conectando uno de los extremos de este interruptor a una entrada digital se puede saber en que momento se abre o se cierra el circuito.



(a) Interruptor Mágnetico (b) Funcionamiento Interruptor Mágnetico

Figura 16: Interruptor Mágnetico

4.4. Actuadores

Un actuador es un dispositivo capaz de transformar energía hidráulica, neumática o eléctrica en la activación de un proceso con la finalidad de generar un efecto sobre un proceso. Estos elementos reciben las órdenes de un regulador o controlador, en este caso, un microcontrolador, y, en función de esta orden, se activa un elemento de control, es decir, el actuador.

Existen varios tipos de actuadores, ya sea, electrónicos o eléctricos. Algunos de los ejemplos de elementos actuadores son, motores, válvulas, LED's o pistones.

En este proyecto en concreto, voy a utilizar dos tipos de actuadores, motores y luces LED. Los cuales, paso a detallar.

* **Micro Servomotor SG90:** Un servomotor o motor paso a paso, es un dispositivo similar a un motor de corriente continua, aunque, a diferencia de este tipo de motores, los servomotores, pueden posicionarse y mantener una posición concreta dentro de su rango de actuación. Dicho rango de actuación, suele ser del orden de 0 a 90 o a 180 grados. Además, un servomotor puede ser controlado tanto en velocidad como en posición.

  Normalmente, se puede modificar un servomotor para convertirlo en un motor de corriente continua, ya que, los servomotores suelen tener una tarjeta electrónica con un potenciómetro que es el elemento que controla el movimiento del motor. Por tanto, si se puentea dicha tarjeta electrónica y se conecta el motor directamente a la corriente continua, el servomotor se convierte en un motor de corriente continua.

  En concreto, para este proyecto, he utilizado varios micro servos **TowerPro SG90**, dicho servo, tiene un rango de funcionamiento de 180º, un peso de 9 gramos, un par motor de 1.8 Kg × cm y un voltaje operativo de 5V.

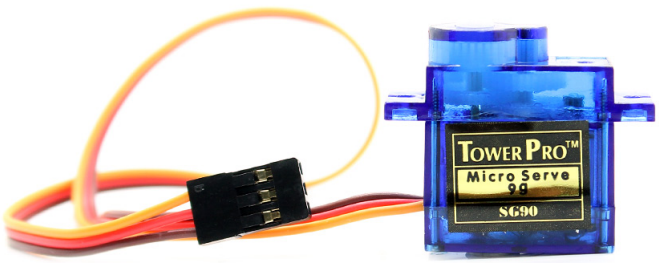


Figura 17: Servomotor TowerPro SG90

* **Servomotor Rotación Continua:** Esta es una variante de los servomotores normales, en los que la señal que se envía al servomotor controla la velocidad y el sentido del giro, y, no la posición como ocurre con un servomotor normal.

  Este tipo de servomotores tiene un rango de actuación de 360º, de ahí el nombre de servomotor de rotación continua.

  En concreto, para este proyecto, he optado por un servomotor de rotación continua bastante simple y barato, este servo es el siguiente, el **FEETECH FS90R**. Las características de este servomotor son, un peso de 9 gramos, un máximo de 130 RPM o revoluciones por minuto y un par motor de 1.5Kg × cm.

Figura 18: Servomotor de Rotación Continua FEETECH FS90R

* **Luces LED:** Los diodos emisores de luz, son los actuadores más básicos que existen. Existen diferentes tipos de LED's, aunque, en este caso, tan solo se van a utilizar LED's de colores.

  Este diodo emisor de luz, se basa en el principio físico que dice que, cuando los pares electron-hueco se mueven para recombinarse, es decir, cuando los electrones caen desde la capa de conducción (la de mayor energía), a la capa de Valencia (la de menor energía), esa pérdida de energía se libera en forma de fotones. Por tanto, el tipo de luz que emita un LED, dependerá de la diferencia de energía entre las capas de conducción y de Valencia, es decir, dependerá del tipo de material que se utilice, ya sea, fosfuro de galio (GaAsP) para emitir luces rojas, anaranjadas o amarillas, fosfuro de Galio (GaP) para luces verdes o seleniuro de cinc (ZnSe) para luces azules.

  Los LED's están polarizados, es decir, solo funcionarán si están conectados correctamente. Normalmente, el ánodo suele ser la patilla más larga del LED, dicho ánodo es la parte positiva del LED. Por tanto, el cátodo es la parte negativa y suele ser la patilla más corta del LED.

  Ilustro aquí los LED que se van a utilizar.

Figura 19: LED's de colores

5 Necesidades del Proyecto

Este proyecto se va a basar en la construcción de una maqueta de una vivienda unifamiliar. Para, mediante el uso de la placa de desarrollo **NodeMCU** y un servidor web programado en esta, controlar diferentes sensores y actuadores que se instalarán en la maqueta.

5.1. Primera Aproximación

Al iniciar este proyecto, mi primera idea para el mismo fue, el intentar utilizar las plataformas **Arduino** y **Raspberry Pi** en conjunción, programando un servidor web en **Raspberry Pi**, visto el hecho que, **Raspberry Pi** tiene mucho más poder de procesamiento que **Arduino**, y, utilizar **Arduino** para controlar los diferentes sensores y actuadores, debido a la facilidad de programación de **Arduino** para el control de estos elementos.

Uno de los primeros problemas de esta solución, fue, la comunicación entre **Arduino** y **Raspberry Pi**, ya que, **Arduino** por sí mismo no integra ningún tipo de módulo de comunicación, ya sea, Ethernet, Wi-Fi o Bluetooth, aunque, se pueden utilizar infinidad de shield's para **Arduino**, que, dotan al mismo de este tipo de comunicaciones. Uno de los shield más utilizados en **Arduino**, es el shield Ethernet, que se encarga de dotar a **Arduino** de un puerto Ethernet para conectarse a una red cableada.

Sin embargo, para este proyecto, quería conseguir una comunicación inalámbrica, ya fuera por Wi-FI, por radiofrecuencia o por Bluetooth, así que, busqué diferentes maneras de conseguir este tipo de conexiones inalámbricas en **Arduino**, encontrando, infinidad de shield's o módulos que dotaban a **Arduino** de este tipo de conexión. Pero, por ejemplo, el shield Wi-Fi oficial para **Arduino**, supera con creces el precio de **Arduino** en sí mismo, por tanto, descarté el uso de este shield para dotar de Wi-Fi a **Arduino**. En el caso de conexión por radiofrecuencia, si existen módulos que trabajan en la frecuencia de los 2.4GHz, con un precio contenido, pero, el problema de estos módulos es, que, tienen un consumo energético bastante elevado y, necesitan una fuente de alimentación estable para conseguir una buena comunicación con un buen alcance.

5.2. Necesidades del Microcontrolador

Las características principales del microcontrolador que se ha de utilizar para este proyecto, tenían que cumplir algunas necesidades básicas, las cuales, paso a enumerar.

* Microcontrolador con varias entradas y salidas digitales y analógicas.
* Reducido tamaño de dicho controlador.
* Posibilidad de conexión inalámbrica del microcontrolador.
* Capacidad de proceso suficiente para programar un servidor web en el microcontrolador.
* Reducido precio del microcontrolador, los sensores y actuadores.

5.3. Necesidades de la maqueta

Por lo que respecta a la construcción y control de los diferentes elementos que componen la maqueta sobre la que se basa el proyecto, esta, había de disponer de suficiente espacio para poder utilizar la placa de desarrollo **NodeMCU**, para el control de los siguientes elementos.

* Controlar luces LED en distintos puntos de la maqueta.
* Controlar servomotores y servomotores de rotación contínua.
* Monitorización de sensor de temperatura.
* Monitorización de sensor de distancia.

Por tanto, vistas las especificaciones y las necesidades del proyecto, se puede ver que la placa **NodeMCU** se ajusta a estas necesidades, siendo, su mayor inconveniente el corto número de pines de entrada/salida de los que dispone.

5.4. Construcción de la Maqueta

La maqueta está construída sobre una base de madera, y cuenta con un comedor y una habitación en un segundo piso, además de, un ascensor, un patio trasero y un garaje. Todo esto, construïdo con cartón pluma, es decir, una lámina de poliestireno en medio de dos láminas de papel estrucado. El uso de este material aporta una cierta rigidez, y, sobre todo, ligereza, ya que, es un material muy liviano que viene muy bien para este tipo de usos.

Una vez realizada la estructura básica, he añadido diferentes elementos para decoración, también fabricados en cartón pluma, tales como, macetas, sofás, camas o electrodomésticos. A su vez, he añadido trozos de césped artificial para la entrada de la maqueta y el patio trasero.

Para la construcción de esta maqueta, he utilizado herramientas muy básicas, tales como, tijeras, un cutter o una pistola de cola caliente.

6 Programación En NodeMCU

En este apartado de la memoria, voy a ilustrar, tanto, la configuración necesaria en el IDE de **Arduino** y diferentes ejemplos básicos de programación en **NodeMCU**, así como, el código fuente completo del proyecto, incluyendo, el código fuente **Arduino** para controlar los pines de entrada/salida de **NodeMCU**, además, del código fuente del servidor web que ha de correr sobre la misma placa para el control de la misma, por medio de la red.

6.1. Instalación Add-on IDE Arduino

Hecho el trabajo previo de investigación y decisión del tipo de placa microcontroladora a utilizar, así como, la construcción de la maqueta y la decisión de los diferentes sensores y actuadores de los que ha de disponer la misma, me pongo a programar y probar los diferentes elementos utilizando la placa **NodeMCU**.

Inicialmente, antes de empezar a programar, es necesario, preparar el IDE de **Arduino** para poder programar la placa **NodeMCU**, ya que, es necesario un add-on para este IDE.

Existen dos opciones para programar **NodeMCU** en **Arduino**, una, descargar un IDE ya configurado con el add-on instalado, o, instalar el add-on por separado. Además, este add-on, no funciona solo para **NodeMCU**, si no, que, permite programar casi cualquier placa que integre el chip ESP8266.

Voy a ilustrar con algunas fotografías, como configurar el IDE de **Arduino** que ya tengo instalado, para utilizar este add-on.

Para poder instalar este add-on, se ha de tener instalada la versión 1.6.4 del IDE o posterior, ya que, a partir de dicha versión el entorno de desarrollo permite crear definiciones de nuevas placas e incorporarlas al entorno. Dicho esto, se ha de descargar el add-on desde los repositorios de **Arduino** de la siguiente manera.

Vamos a \Archivo\Preferencias y, nos aparecerá una ventana como la siguiente, dónde, en la parte inferior, en Gestor de URLs Adicionales de Tarjetas, se tendrá que escribir el siguiente vínculo para que el entorno sea capaz de descargar el add-on que se necesita.

Link: http://arduino.esp8266.com/stable/package_esp8266com_index.json

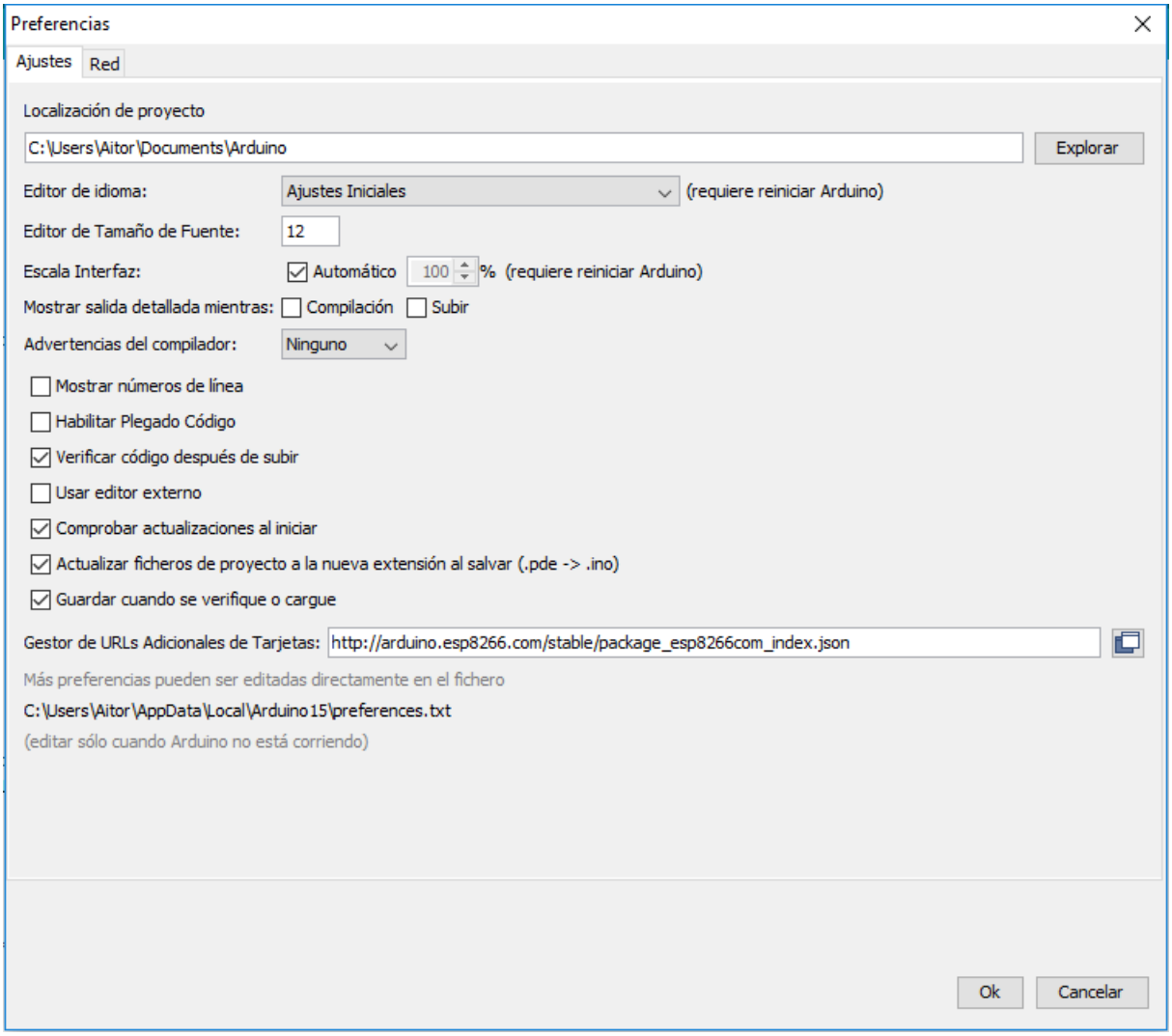


Figura 20: Preferencias

Una vez hecho click en OK, vamos a \Herramientas\Placa\Gestor de Tarjetas

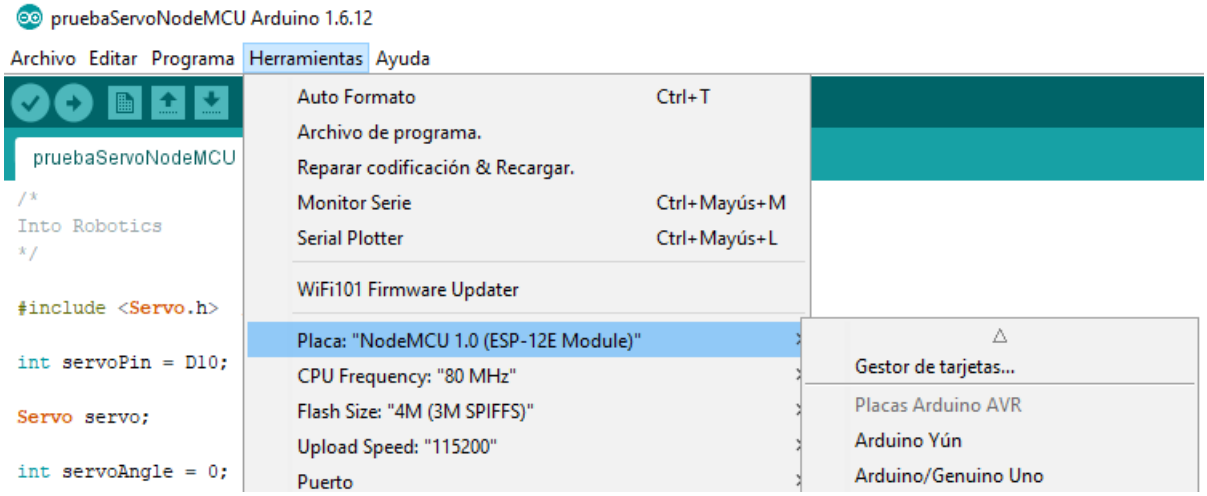


Figura 21: Gestor de Tarjetas

Posteriormente, en el apartado de gestor de tarjetas, se busca la placa esp8266 Community Version, y se hace click en instalar, una vez hecho esto, aparece una barra de progreso que irá llenándose a medida que se descarguen los archivos necesarios.

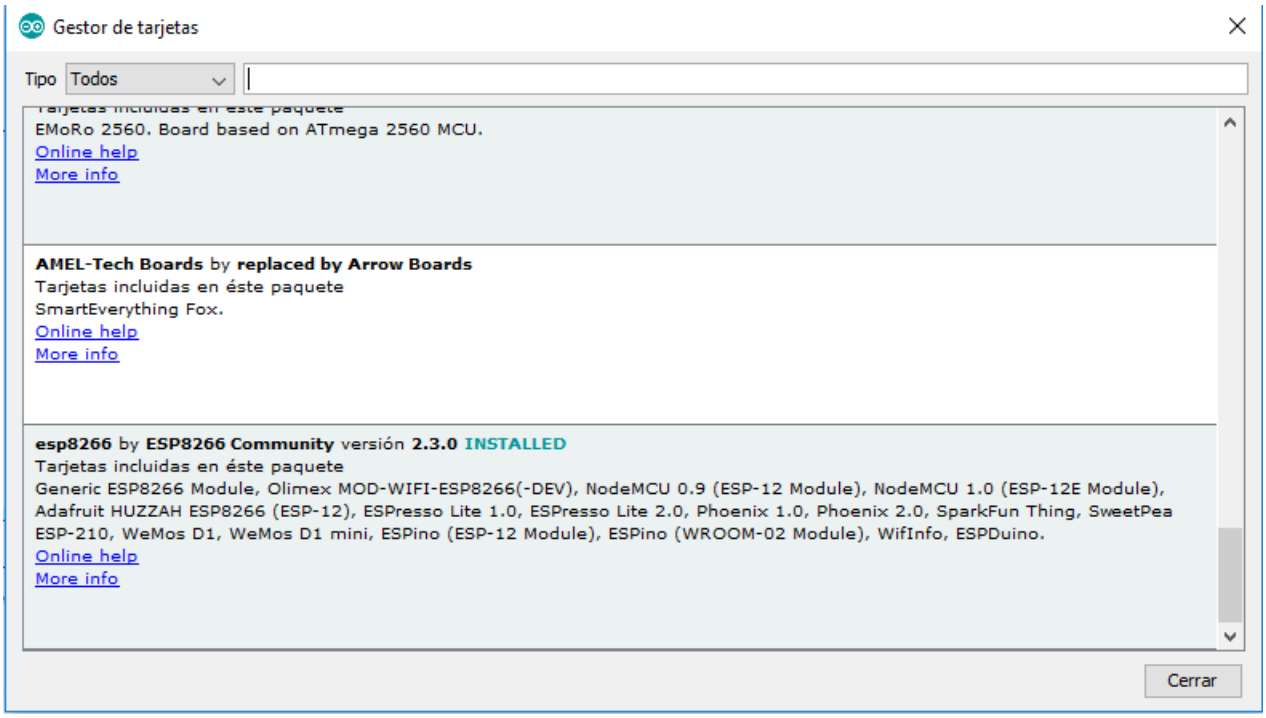


Figura 22: Instalación Add-On

Finalizado este paso, en \Herramientas\Placa aparecerán multitud de placas de desarrollo que integran el ESP8266, entre ellas, la placa **NodeMCU** 1.0 que se necesita para este proyecto.

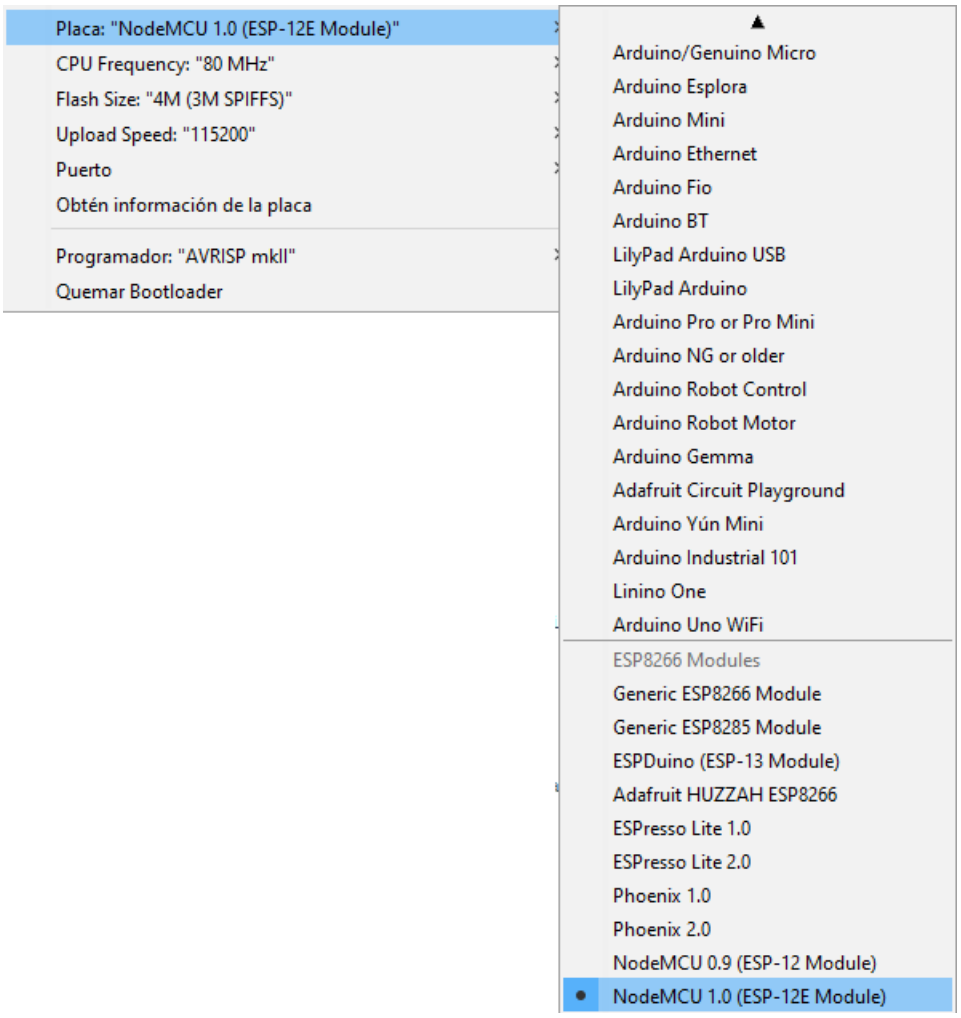


Figura 23: Selección Placa NodeMCU

En este punto, el add-on ya estará instalado en el IDE de **Arduino**, y la placa de desarrollo que se utilizará para este proyecto seleccionada, aunque, para una correcta comunicación entre el PC dónde está instalado el IDE de **Arduino** y la placa **NodeMCU**, se ha de configurar la velocidad de subida del "Sketch" o programa, desde el PC a la placa de desarrollo. Normalmente, para proyectos con placas **Arduino**, esta velocidad es de 9600 baudios por defecto, pero, para esta placa, se ha de configurar la velocidad a 115200 baudios. Para esto, en \Herramientas\Upload Speed se marca la opción 115200.

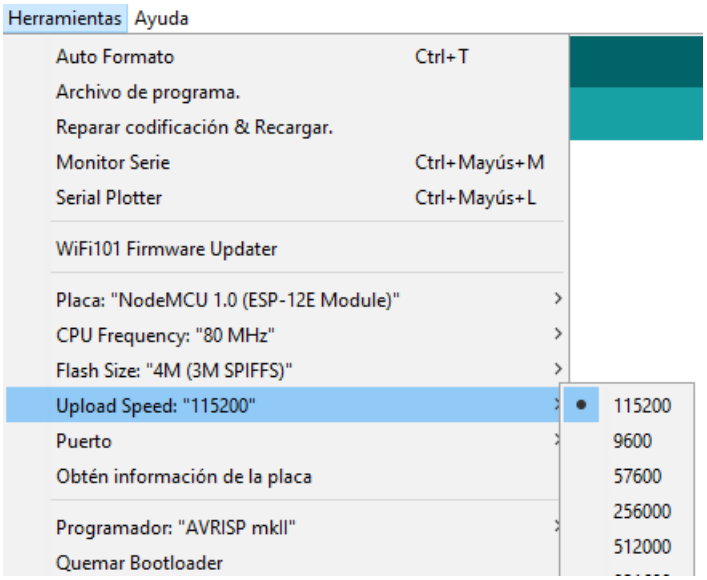


Figura 24: Velocidad de Subida del Programa a NodeMCU

Una vez hechos estos ajustes al IDE de **Arduino**, ya se puede programar la placa **NodeMCU** utilizando el mismo lenguaje **Arduino**, facilitando así el uso de esta placa, ya que, este IDE es mucho más intuitivo que programar la placa en lenguaje LUA.

6.2. Ejemplos de Programación NodeMCU

Primero, empiezo con las pruebas más básicas de programación sobre placas de desarrollo, dónde se enmarcan pruebas, tales como, encender un led, o monitorizar la entrada o salida analógica o digital que proporciona un sensor, más tarde, pasando a controlar los servomotores y servomotores de rotación continua de los que dispone este proyecto. Durante este punto de la memoria voy a ilustrar con código fuente **Arduino**, algunos ejemplos de control o monitorización de dichos sensores y actuadores, para, en un los puntos posteriores, ilustrar el código fuente del servidor web necesario para controlar los elementos conectados a la placa de desarrollo **NodeMCU**.

Dicho esto, paso a exponer diferentes ejemplos de código fuente para el control básico de sensores y actuadores.

Al igual que en cuando se programa en **Arduino**, el código para la placa **NodeMCU**, está dividido en dos partes, la primera parte de código se llama "setup" y la segunda se llama "loop". Siendo la primera parte del código la que se ejecuta en el momento del encendido o reprogramación de la placa, y, la segunda parte del código se va ejecutando en un bucle infinito mientras la placa está encendida, en este caso, el "loop".

Control Básico de un LED

```
1 int pinLed = 13;
2
3 void setup(){
4     pinMode(pinLed, OUTPUT);
5 }
6
7 void loop(){
8     digitalWrite(pinLed,HIGH);
9 }//End
```

Listing 6.1: Control Básico Led

Este ejemplo, que, viene a ser un "Hola Mundo" para la programación de placas de desarrollo, ya que, suele ser el primero que se escribe al utilizar una nueva placa de desarrollo. Dicho código, muestra las directrices necesarias para controlar un diodo emisor de luz conectado al pin número 13 de la placa de desarrollo.

Es decir, en un primer momento, una de las buenas prácticas al programar en placas de desarrollo es crear una variable que contenga el valor nominal del pin al que se ha conectado el dispositivo a utilizar o controlar, en este caso, un led en el pin 13, ya que, si, más adelante en el código, se quisiera cambiar de pin dicho dispositivo, tan solo se tendría que cambiar el valor de esta variable y no todos los valores de las llamadas a funciones que impliquen este elemento.

Dicho esto, en este código en concreto, se establece, mediante una llamada a la función **pinMode**, que, el led conectado al pin 13 de la placa, ha de actuar como **OUTPUT**, es decir, como una salida. Establecido dicho pin como una salida, mediante la llamada **digitalWrite**, se establece a **HIGH** este pin, es decir, este pin pasa a estar activado o a tener un valor alto, encendiendo así el led conectado al mismo.

Control Servomotor

```
#include <Servo.h>

int servoPin = 10;
Servo servo;

//Posicion del Servomotor en grados
int servoAngle = 0;

void setup()
{
  servo.attach(servoPin);
  servo.write(0);
}


void loop()
{
    //Controlar la velocidad del servo
    //Si se cambia el valor del delay, cambia la velocidad de rotacion del servo
    //Bucle for para mover el servo, desde 0 grados a 180 grados

    for(servoAngle = 0; servoAngle < 180; servoAngle++)
    {
        servo.write(servoAngle);
        delay(10);
    }
```

```
27
28      //Ahora, de vuelta a 0 grados desde 180 grados
29      for(servoAngle = 180; servoAngle > 0; servoAngle--)
30      {
31          servo.write(servoAngle);
32          delay(10);
33      }
34  }//End
```

Listing 6.2: Control Servomotor

En este ejemplo, se muestra la forma de controlar un servomotor, para, en este caso, realizar un barrido de 0º a 180º, para luego volver a 0º. Para completar esta tarea, se utilizará una librería incluida en el IDE **Arduino**, la cual, se denomina, Servo.h, y, para incluirla en el código, se ha de utilizar la cláusula #include.

Una vez incluida la librería en el código, tal y como se ha explicado en el ejemplo anterior, se declara una variable que contenga el valor del pin de entrada/salida al que está conectado el servomotor, en este caso, el pin 10. Hecho esto, se declara un objeto Servo, de nombre servo, en este caso, para, posteriormente, en el setup del "Sketch", "atar" este objeto Servo al pin donde estará conectado el servomotor, mediante la llamada attach(). Una vez atado el objeto Servo al pin donde está conectado, mediante la función write(), se le envía al servomotor el grado en el que se ha de posicionar, en este caso, en el setup, la posición inicial del servomotor será la de 0º.

Una vez finalizados los preliminares del programa, en el loop, se realiza la acción principal, en este caso, utilizando dos bucles for(), se realiza un barrido desde 0º a 180º, mediante una llamada a la función write(), además de una variable contador (servoAngle) para el bucle, para, una vez en la posición de 180º, con otro bucle for() volver a la posición 0º.

Control Sensor de Distancia

```
1  const int echoPin = 8;
2  const int triggerPin = 9;
3
4  void setup() {
5
6      //Inicio de la comunicacion Serial
7      Serial.begin(9600);
8
9      //Activacion del pin 9 como salida, para el pulso ultrasonico
```

```
10      pinMode(triggerPin,OUTPUT);
11
12      //Pin 8 como entrada, tiempo de rebote del sonido
13      pinMode(echoPin,INPUT);
14  }
15
16  void loop() {
17
18      int cm = ping(triggerPin,echoPin);
19      Serial.print("Distancia: ");
20      Serial.println(cm);
21
22      delay(1000);
23
24  }
25
26  int ping(int triggerPin, int echoPin){
27
28      long duration,distanceCm;
29
30      //Para generar un pulso limpio
31      digitalWrite(triggerPin, LOW);
32      delayMicroseconds(4);
33
34      //Generamos senal
35      digitalWrite(triggerPin, HIGH);
36      delayMicroseconds(10);
37      digitalWrite(triggerPin,LOW);
38
39      //Medimos el tiempo entre pulsos
40      duration = pulseIn(echoPin, HIGH);
41
42      //Convertimos distancia a cm
43      distanceCm = duration * 10/ 29.2 /2;
44
45      return distanceCm;
46  }
```

Listing 6.3: Control Sensor de Distancia

En este ejemplo, se ilustra como se puede controlar un sensor de distancia, el cual, se basa en el uso de ultrasonidos, ya que, consta de un emisor y un receptor de ultrasonidos, los cuales, mediante la emisión de ultrasonidos, permite detectar objetos gracias a, que, cuando la señal del ultrasonido rebota en un objeto, el receptor del sensor la detecta, y, una vez detectada esta seña, se puede calcular la distancia a la que se encuentra dicho objeto.

Una vez explicado su funcionamiento físico, es más fácil entender el código para utilizar este sensor, primero, se declaran la variables con el valor nominal de los pines a los que está conectado el sensor, en este caso, este sensor utiliza dos pines, uno para el emisor y otro para el receptor, una vez declarados los pines, en el setup, se inicia la comunicación por el puerto serial, para poder mostrar por el monitor serie la distancia a la que se encuentra el objeto, además, se declaran, el pin 9 como salida para emitir ultrasonidos, y el pin 8 como entrada para el receptor de ultrasonidos.

Hecho esto, para claridad del código, se declara una función propia, para calcular la distancia a la que se encuentra un objeto. Esta función recibe como argumentos, los pines a los que está conectado el sensor, y, se declaran dos variables **long**, para medir el tiempo entre pulsos ultrasónicos y para calcular la distancia en centímetros al objeto (duration, distanceCm). Declaradas estas variables, mediante la llamada a la función digitalWrite(), primero, se pone a LOW el emisor para generar una señal limpia, posteriormente, después de un delay de 4 microsegundos, se genera la señal poniendo a HIGH el emisor, esperando 10 microsegundos, y, poniendo a LOW el emisor, esto, provoca que el emisor envíe pulsos ultrasónicos de 10 microsegundos de duración. Una vez generada la señal, se mide el tiempo entre los pulsos con el receptor del sensor, asignando el valor que devuelve la llamada a la función pulseIn a la variable duration, declarada anteriormente. Esta función pulseIn(), en este caso, dispone de dos argumentos, uno de ellos, el pin donde está conectado el receptor del sensor, y el segundo, el valor que se ha de esperar de este pin para que se active la función, es decir, cuando el pin tiene un valor HIGH, se inicia una cuenta temporal, finalizando dicha cuenta en el momento que el pin cambia a LOW, una vez realizada la cuenta, la función devuelve el valor de la misma, y, en este caso, este valor, se asigna a la variable duration.

Para el cálculo de la distancia en cm a la que se encuentra un objeto, se utilizan varias fórmulas y suposiciones, ya que, se sabe que la velocidad del sonido es de 343 metros/segundo, por tanto.

$$343 m/s * 100 cm/m * \frac{1}{1000000} s/\gamma s = \frac{1}{29{,}2} cm/\gamma s$$

Es decir, el sonido tarda 29,2 microsegundos en recorrer un centímetro, por tanto, se puede obtener la distancia, a partir de, la duración del tiempo entre la emisión y la recepción de un pulso ultrasónico.

$$Distancia(cm) = \frac{Tiempo(\gamma s)}{29{,}2 * 2}$$

Se divide el tiempo total entre dos, ya que, se mide el tiempo que tarda el pulso en ir y volver, por lo que, la distancia que recorre este pulso, es el doble de la que queremos conocer, en este caso, la distancia a la que se encuentra el objeto.

Una vez visto esto, en este caso concreto, para calcular la distancia a la que se encuentra un objeto, la fórmula que se utiliza es la siguiente:

$$distanceCm = \frac{duration * 10}{\frac{29{,}2}{2}}$$

Programada esta función, en el loop del programa, se declara una variable de tipo entero (cm), donde se guarda el valor que se devuelve al llamar a la función ping(). Hecha esta llamada, se imprime el valor de esta variable entera en el monitor serie, mediante la función print.

Correspondencia pines NodeMCU con pines Arduino

Uno de las diferencias más claras entre la placa de desarrollo **NodeMCU**, y, las diferentes placas de desarrollo **Arduino**, es la diferencia en la numeración de los pines de entrada/salida.

Así que, para programar **NodeMCU** completamente como un **Arduino**, en el preámbulo del "sketch", se puede hacer uso de la cláusula #define, para definir como constantes, la correspondencia entre los pines de **NodeMCU** y los pines de **Arduino**, dicha correspondencia se puede observar en el siguiente extracto de código fuente.

En dicho extracto, la correspondencia es la siguiente, los pines correspondientes a **Arduino**, serán los nombrados justo después de la declaración de la cláusula #define, para, nombrar los pines correspondientes a la placa de desarrollo **NodeMCU** justo después de los pines de **Arduino**. Por tanto, la definición de constantes quedaría de la siguiente manera, **#define PinArduino PinNodeMCU**.

Dicho esto, si se incluye el siguiente extracto de código, en el preámbulo del "sketch" del IDE **Arduino**, se pueden portar ejemplos de código, programados para **Arduino**, que funcionen sobre la placa de desarrollo **NodeMCU**.

```
/*Asignacion de pines de Arduino a los pines GPIO de NodeMCU LoLin*/
#define A0 A0

#define D0 16
#define D1 5
#define D2 4
#define D3 0
#define D4 2
#define D5 14
#define D6 12
#define D7 13
#define D8 15
#define D9 3
#define D10 1
```

Listing 6.4: Correspondencia Pines NodeMCU-Arduino

6.3. Código Fuente Control NodeMCU

Una vez explicados algunos ejemplos de programación sobre **NodeMCU**, así como, la correspondencia de los pines de **NodeMCU** v3 LoLin con los pines de **Arduino** Uno, en esta sección voy a ilustrar, comentar y explicar el código fuente, programado mediante el IDE de **Arduino**, necesario para controlar los pines de entrada/salida de **NodeMCU**, a los que están conectados los diferentes sensores y actuadores de los que dispone la maqueta.

En esta primera sección de código, se va a ver las librerías de las que hace uso el código, así como, las cláusulas #define que marcan la correspondencia de los pines ya comentada.

```
/*Librerias*/
#include <ESP8266WiFi.h>
#include <Servo.h>

/*Asignacion de pines de Arduino a los pines GPIO de NodeMCU LoLin*/
#define A0 A0

#define D0 16
#define D1 5
#define D2 4
#define D3 0
#define D4 2
#define D5 14
```

```
#define D6 12
#define D7 13
#define D8 15
#define D9 3
#define D10 1
```

Listing 6.5: Librerías Utilizadas y Correspondencia Pines NodeMCU-Arduino

Así pues, las librerías, en este caso, la <ESP8266WiFi.h>y la <Servo.h>, permiten el uso de las funciones inalámbricas de la placa de desarrollo **NodeMCU**, así como, el control de los servomotores de los que dispone la maqueta.

Visto esto, una vez cargada la librería que permite el control del SoC Wi-Fi ESP8266, mediante dos punteros de tipo char, se establecen dos constantes, el SSID o Nombre de Red y la contraseña de la red a la que se debe conectar dicho SoC.

```
//Conectar NodeMCU a red Wi-Fi
const char* ssid = "SSID de la Red";
const char* password = "Password de la Red";
```

Listing 6.6: Conexión NodeMCU Red Wi-Fi

Por tanto, en este punto del código, la placa de desarrollo **NodeMCU**, ya dispondría de los elementos necesarios para conectarse a la red Wi-Fi deseada, es en este momento, cuando se puede empezar a declarar variables que contengan los valores nominales de los pines de **NodeMCU**, y, asignar estos valores a las diferentes variables para controlar, tanto los sensores, como los actuadores

```
/*DECLARACION VARIABLES CONTROL PINES*/
//Declaracion Pines Luces
int pinLedEntrada = 16;                          //GPIO 16 Led Entrada
int pinLedCocina = 5;                            //GPIO 05 Led Cocina Comedor

//Declaracion Pin Aire Acondicionado
int pinLedAC = 4;                                //GPIO 04 Led Habitacion

//Declaracion Pines Motores
int pinMotorGaraje = 3;                          //GPIO 00 Motor Cochera
int pinMotorEntrada = 2;                         //GPIO 02 Motor Puerta Entrada
int pinMotorVentilador = 14;                     //GPIO 14 Motor Ventilador
int pinMotorPersianasAbajo = 12;                 //GPIO 12 Motor Persianas Abajo
int pinMotorPersianasArriba = 13;                //GPIO 13 Motor Persianas
    Arriba
```

```
15
16 //Declaracion Pines Sensores
17 int pinSensorPuertaPrincipal = 15;                    //GPIO 15 Sensor Magnetico
       Puerta Principal
18 int triggerPinSensorCochera = 0;                     //GPIO 03 Trigger Sensor
       Distancia Puerta Cochera
19 int echoPinSensorCochera = 1;                       //GPIO 01 Echo Sensor Distancia
        Puerta Cochera
20 int pinSensorTemperatura = A0;                      //A0 Sensor Temperatura
```

Listing 6.7: Declaración Variables de Control de los Pines de NodeMCU

Como se ha comentado anteriormente, se trata de una buena práctica, realizar la declaración de estas variables, para, facilitar un futuro cambio de pines en sensores y actuadores.

Se puede observar, que, el único pin de entrada analógico de **NodeMCU**, se va a utilizar para conectar un sensor de temperatura que permita conocer la temperatura interior de la maqueta, ya que, el conocimiento de la temperatura, es, básico, para el control de la climatización o refrigeración de un hogar domótico.

Después de declarar las variables para el control de los pines de entrada/salida, se puede pasar a declarar los objetos y variables necesarios para controlar los servomotores de los que dispone la maqueta, así pues, mediante el siguiente código, se establecen los objetos necesarios para el control de los servomotores para la puerta del garaje, para la puerta de entrada, así como, los motores para las persianas de la maqueta. Al mismo tiempo, se declaran las variables de control para conocer y controlar la posición de dichos servomotores.

```
1 /*DECLARACION OBJETOS */
2 Servo servoGaraje;
3 Servo servoEntrada;
4 Servo servoPersianaPisoAbajo;
5 Servo servoPersianaPisoArriba;
6 Servo servoVentilador;
7
8 int posicionServoGaraje;
9 int posicionServoEntrada;
10 int posicionServoPersianaPisoAbajo;
11 int posicionServoPersianaPisoArriba;
12 int posicionServoVentilador;
```

Listing 6.8: Declaración Objetos y Variables de Control de los Servomotores

Como se ve, se declara el objeto mediante la cláusula Servo, contenida en la librería Servo incluida anteriormente, dándole el nombre neceario a estos objetos, a su vez, se declaran variables de control, de tipo int, para conocer y controlar la posición de los diferentes servomotores declarados.

En este punto del programa, ya están declarados los objetos y las variables de control que se van a necesitar en el programa, aunque, antes de empezar el "setup" del programa, se ha de declarar el puerto por el que se ha de comunicar el usuario con el servidor web programado en **NodeMCU**, en este caso, el puerto 80, dicho puerto, se trata del predefinido para que el servidor web HTTP escuche las peticiones de los clientes que se conectan a este servidor.

```
1 //Declaracion del puerto 80 como puerto de escucha del servidor
2 WiFiServer server(80);
```

Listing 6.9: Declaración Puerto de Escucha del Servidor Web

La declaración de este puerto se realiza mediante la cláusula WiFiServer incluida por defecto en el IDE de **Arduino**, seguida, esta cláusula, de, server(puerto), en este caso, el puerto 80.

Con esta declaración, termina el preámbulo del "sketch", y, empieza el "setup" del programa, en el cual, para empezar, se ha de inicializar la comunicación serial, para poder imprimir mensajes en el monitor del IDE de **Arduino**, a modo de "debugging", además, son necesarias tantas llamadas a la función attach() para la fijación de los pines elegidos en el preámbulo, como pines para los servomotores de la maqueta se utilizen. Al mismo tiempo, se inicializan a 0 las variables de control de la posición de los servomotores, principalmente, para que los servomotores se inicialicen en una posición fija común. Una vez hecho esto, se han de declarar los modos para los pines de entrada/salida, llamando a la función pinMode() incluida en el IDE de **Arduino**.

Posteriormente, para finalizar el "setup" se realizan las llamadas a las funciones necesarias para conectar la placa de desarrollo a la red Wi-Fi deseada e iniciar el servidor web programada en dicha placa.

```
1  void setup(){
2
3   //Iniciacion del puerto serial
4   Serial.begin(115200);
5   delay(10);
6
7   //Ligar los pines a los servomotores
8   servoGaraje.attach(pinMotorGaraje);
9   servoEntrada.attach(pinMotorEntrada);
10  servoPersianaPisoAbajo.attach(pinMotorPersianasAbajo);
```

```
11 servoPersianaPisoArriba.attach(pinMotorPersianasArriba);
12
13 //Inicializacion a 0 de las variables de control de los servomotores
14 posicionServoGaraje = 0;
15 posicionServoEntrada = 0;
16 posicionServoPersianaPisoAbajo = 0;
17 posicionServoPersianaPisoArriba = 0;
18
19 //Posicionar servomotores a 0
20 servoGaraje.write(posicionServoGaraje);
21 servoEntrada.write(posicionServoEntrada);
22 servoPersianaPisoAbajo.write(posicionServoPersianaPisoAbajo);
23 servoPersianaPisoArriba.write(posicionServoPersianaPisoArriba);
```

Listing 6.10: Inicio del Setup del Sketch

Es en este fragmento del código, donde, se inicializa la comunicación Serial, mediante Serial.begin(), en este caso, a 115200 baudios, ya que, es esta velocidad la que demanda la placa de desarrollo, una vez inicializada, se pasa a ligar los pines a los que están conectados los servomotores, con los objetos Servo declarados en el código, todo esto, mediante la llamada a la función attach(), perteneciente esta a la librería Servo que se ha inicializado anteriormente. Una vez ligados los servomotores, ya se puede controlar su posición, esto es posible mediante la llamada a la función write(), también incluida en la librería añadida, además de, un argumento para esta función, el cual, puede ser directamente el valor nominal del grado al que se quiere que llegue el servomotor, o, una variable que contenga este valor, siendo esta segunda opción, la más recomendable, ya que, permite un mayor control y facilidad del código, en este caso, se utiliza esta opción al inicio del "setup", para, en caso de iniciar o reiniciar la placa **NodeMCU**, forzar a los servomotores a moverse a una posición incial común y conocida, en este caso, 0º.

Controlados ya los servomotores normales, para el control del servomotor de rotación continua del que hará uso la maqueta, se usa un código muy similar al del control de los servomotores normales, siendo, la única diferencia el valor de los grados que hay que utilizar, ya que, si se desea detener la rotación del servomotor, se ha de establecer un valor de 90, un valor de 0 para el movimiento en sentido de las agujas del reloj, y, un valor de 180 para el movimiento contrario, es decir, contrario a las agujas del reloj.

```
servoVentilador.attach(pinMotorVentilador);

posicionServoVentilador = 90;      //Detenido

servoVentilador.write(posicionServoVentilador);
```

Listing 6.11: Control Servomotor Rotación Continua

Iniciando así el programa, con el servomotor de rotación contínua, detenido.

Siguiendo con el código, una vez controlados los servomotores, se pasa a declarar el modo en el que van a actuar los diferentes pines de entrada/salida a los que están conectados los sensores y actuadores necesarios, es decir, se ha de establecer si un pin es de entrada, o es de salida.

```
/*PinMode del sensor de TEMPERATURA*/
pinMode(pinSensorTemperatura, INPUT);

/*PinMode del sensor MAGNETICO de la PUERTA PRINCIPAL*/
pinMode(pinSensorPuertaPrincipal, INPUT);

/*PinMode del sensor de DISTANCIA de la puerta GARAJE*/
pinMode(triggerPinSensorCochera, OUTPUT);
pinMode(echoPinSensorCochera, INPUT);

/*PinMode de los LED*/
pinMode(pinLedEntrada, OUTPUT);
pinMode(pinLedCocina, OUTPUT);

digitalWrite(pinLedEntrada,LOW);
digitalWrite(pinLedCocina,LOW);

/*PinMode del Led del Aire Acondicionado*/
pinMode(pinLedAC, OUTPUT);
digitalWrite(pinLedAC,LOW);
```

Listing 6.12: Declaración de los Modos de los Pines

Para establecer un pin como entrada o salida, se ha de utilizar la función pinMode, incluida en el IDE **Arduino**, pasándole como argumentos a dicha función, en primer lugar, el nombre de la variable que contenga el valor nominal del pin, o, directamente el valor nominal del pin, en este caso, por claridad, se utiliza una variable, acto seguido, en mayúscula, se ha de establecer el modo del pin, INPUT para un pin de entrada o OUTPUT para un pin de salida. En este caso concreto,

el pin al que está conectado el sensor de temperatura ha de actuar como un pin de entrada, y, los pines a los que están conectados los LED's que se utilizan en la maqueta, han de actuar como salidas.

Otra buena práctica en la programación sobre placas de desarrollo, es, el hecho de que, al momento que se declara el modo al que pertenece un pin al que se conecta un LED, se realiza una llamada a la función digitalWrite(), pasándole la variable que contenga el valor del pin, así como, LOW como argumentos, para forzar, que, dicho LED se encuentre apagado en el momento que se inicia o reinicia la placa de desarrollo y vuelve a cargar el programa.

En este estado, el programa ya tiene todos los sensores y actuadores inicializados para poder manipularlos u obtener información de los mismos, y, para finalizar el "setup", se ha de programar la conexión a la red Wi-Fi con SSID y contraseña ya conocidos, además de, iniciar el servidor web, el código del cual se ha de incluir en el "loop" del "sketch" **Arduino**.

```
1  //Conectar al WiFi
2  Serial.println();
3  Serial.println();
4  Serial.print("Conectando a ");
5  Serial.println(ssid);
6
7  WiFi.begin(ssid,password);
8
9  while(WiFi.status() != WL_CONNECTED){
10  delay(500);
11  Serial.print(".");
12 }
13
14 Serial.println("");
15 Serial.println("WiFi Conectado");
```

Listing 6.13: Conexión a la red Wi-Fi Deseada

Es en este fragmento, en el cual, se conecta la placa de desarrollo **NodeMCU** a la red Wi-Fi a la que se desee conectar. En primer lugar, se imprimen por el monitor serial, una serie de elementos que indiquen que la conexión se va a realizar, en este caso, dos líneas en blanco, un mensaje de "Conectando a ", seguido del SSID de la red a la que se va a conectar la placa, siendo esta SSID, la configurada en el preámbulo del programa en la variable ssid. Impreso este mensaje, mediante la llamada a la función WiFi.begin(), con las variables ssid y password como argumentos, se realizar la conexión a la red Wi-Fi, recordando que, esta función esta incluida en la librería ESP8266WiFI.

Hecha la llamada a la función begin(), en este caso, se inicia un bucle para la detección de un error de conexión, en este caso, un bucle while, que, imprime por pantalla el caracter "." cada 500 milisegundos, mientras el estado de la conexión, conocido mediante la llamada a la función WiFi.status(), sea diferente del valor WL_CONNECTED, para, en el momento que WiFi.status() sea igual a WL_CONNECTED, el bucle cumpla su condición de salida y se imprima por el monitor serial el mensaje "WiFi Conectado", haciendo así al usuario, conocedor que **NodeMCU** se ha conectado con éxito a la red Wi-Fi deseada.

Es momento este, para iniciar el servidor web en la placa de desarrollo **NodeMCU**, y, para tal objetivo, se ha de realizar una llamada a la función begin() desde el objeto server creado anteriormente.

```
1 //Iniciar el Servidor
2 server.begin();
3 Serial.println("Servidor Iniciado");
```

Listing 6.14: Inicialización del Servidor Web

Una vez iniciado el servidor, en este caso, se imprime la dirección IP que el router otorga a **NodeMCU**, para poder controlar dicho servidor web, es decir, para poder controlar el sistema desde cualquier equipo conectado a la misma red que disponga de navegador web.

```
1 //Printear la direccion IP
2 Serial.print("Usa esta URL para conectarte: ");
3 Serial.print("http://");
4 Serial.print(WiFi.localIP());
5 Serial.println("/");
6
7 }//Final setup
```

Listing 6.15: Dirección IP Local del Servidor Web

Mostrado el mensaje de inicio con la dirección IP que se otorga a **NodeMCU**, finaliza el "setup" del "sketch", y, se empieza el "loop", es decir, el programa principal que se va a encargar del correcto funcionamiento del sistema, así como, de la parte del código dónde se va a programar el servidor web.

Para empezar esta parte del código fuente, se realizan las llamadas a la función de lectura de los pines de entrada/salida, tanto analógicos como digitales, en este caso, se llama a la función digitalRead() y analogRead(), para los pines digitales y los analógicos, respectivamente, asignando los valores de estas lecturas a diferentes variables contenedoras, en este caso, de tipo int o entero.

```
/*LECTURA DE PINES DE ENTRADA/SALIDA*/

/*Lectura Pines LED*/
int valorLedCocina = digitalRead(pinLedCocina);
int valorLedEntrada = digitalRead(pinLedEntrada);

/*Lectura Pin Aire Acondicionado*/
int valorLedAC = digitalRead(pinLedAC);

/*Lectura Pin Termometro*/
float lectura = analogRead(pinSensorTemperatura);
//float millivolts = (lectura/1023.0)*5000;
float celsius = (3.3* lectura *100.0)/1024;
delay(1000);

/*Lectura Sensor Magnetico Puerta Principal*/
int puertaPrincipal = digitalRead(pinSensorPuertaPrincipal);

/*Lectura Sensor Distancia Puerta Garaje (Mediante Funcion Auxiliar)*/
int cm = ping(triggerPinSensorCochera,echoPinSensorCochera);
```

Listing 6.16: Lectura de los pines de Entrada/Salida

Se observa así, que, se lee, en este caso, los valores de los pines de los LED de la cocina y la entrada, así como, del LED indicador del estado del aire acondicionado. A su vez, mediante la lectura analógica, y la posterior conversión de este valor, se conoce la temperatura ambiente entregada por el sensor de temperatura. Este código se ejecutará en forma de bucle mientras la placa de desarrollo esté alimentada, dejando un valor temporal de 1 segundo entre lecturas, gracias a la llamada a la función delay().

Se observa en el fragmento de código para la lectura de los pines, que, la lectura de los pines del sensor de distancia de la puerta del garaje, se realiza mediante una llamada a la función ping(), pasándole como argumentos, los pines a los que están conectados el emisor y el receptor de ultrasonidos del sensor, el código de dicha función se ha de colocar al acabar el "loop" del "sketch", y, en esta función auxiliar, se generan las señales necesarias para conocer la duración del tiempo de rebote de la señal ultrasónica, para, así, convertir dicha duración a un valor de distancia en centímetros.

```
/*Funcion Auxiliar Sensor Distancia*/

int ping(int triggerPinSensorCochera, int echoPinSensorCochera){

  long duration,distanceCm;

  digitalWrite(triggerPinSensorCochera, LOW); //Generacion de un pulso limpio
  delayMicroseconds(4);
  digitalWrite(triggerPinSensorCochera, HIGH); //Generamos senal
  delayMicroseconds(10);
  digitalWrite(triggerPinSensorCochera,LOW);

  duration = pulseIn(echoPinSensorCochera, HIGH); //Medimos el tiempo entre pulsos

  distanceCm = duration * 10/ 292 /2; //Convertimos distancia a cm
  return distanceCm;

}
```

Listing 6.17: Función Auxiliar Cálculo Distancia

6.4. Código Fuente Servidor Web NodeMCU

En esta sección voy a explicar el código programado para el Servidor Web que se encargará de la gestión de los diferentes actuadores del sistema, dicho Servidor Web, estará programado y se ejecutará en la misma placa de desarrollo **NodeMCU** a la que están conectados los diferentes sensores y actuadores de los que dispone el sistema.

Como primer paso, se ha de conocer si existe algún cliente conectado al servidor web, para, en caso negativo, esperar a que un cliente envíe algún dato, para esto, se crea una instancia WiFiClient, llamada client, y, se le asigna el valor de retorno de la llamada a la función available() sobre el servidor, valor de retorno, que, en este caso, es un tipo de dato cliente, propio de la librería server de **Arduino**. Una vez asignado el valor cliente a client, mediante una cláusula if, dándole como condición que no exista un cliente que se quiera conectar, el servidor queda en estado de escucha a la espera de que la condición de la cláusula if no se cumpla, es decir, que exista un cliente que se quiera conectar al servidor.

```
/*CODIGO DEL SERVIDOR WEB*/

  //Comprobar si hay algun cliente conectado
  WiFiClient client = server.available();

  if(!client){
    return;
  }

  //Esperar hasta que el cliente envie algun dato
  Serial.println("new client");
  while(!client.available()){
    delay(1);
  }

  //Leer la primera linea de la peticion
  String request = client.readStringUntil('/r');
  Serial.println(request);
  client.flush();
```

Listing 6.18: Lectura de los pines de Entrada/Salida

Una vez se cumple la condición del if, es decir, que exista un cliente conectado al servidor, se imprime por pantalla la existencia de este cliente, y, mediante un bucle While, se espera a que este cliente envíe algún dato al servidor, siendo necesaria, para la salida del bucle, que, la llamada a la función available() sobre la instancia client, devuelva el número de bytes que están disponibles para ser leídos desde el servidor.

Hecho esto, se pasa a leer línea por línea la petición del cliente, mediante la función readStringUntil() ejecutada sobre el cliente, y, con el carácter de retorno de carro /r como terminación de la lectura, siendo este necesario para la lectura línea por línea. Además, las líneas se leen gracias a que se asignan a la variable de tipo String, request, y mediante la llamada a la función flush(), se espera a que el buffer que contiene la petición del cliente se vacíe por completo.

Terminada de leer la petición, se pasa a crear diferentes cláusulas if que permitan activar o desactivar los diferentes actuadores, dependiendo de las peticiones que hayan realizado los clientes, para esto, se han de utilizar las variables contenedoras del valor de los pines a los que están conectados los elementos, dichas variables se han declarado en el preámbulo del "sketch". Además de estas variables, se ha de utilizar la llamada a la función digitalWrite() para cambiar el estado de los diferentes pines, dependiendo de la petición del cliente.

```
/*Si el led de la cocina o entrada, esta apagado, y, desde la web se pide
    encenderlo, se cambia el estado del pin a HIGH.

  Si estan encendidos y se pide apagarlos desde la web, se cambia el estado del
    pin a LOW.*/

  if(valorLedCocina == LOW && request.indexOf("/LED_COCINA=ON") != -1){

    digitalWrite(pinLedCocina,HIGH);
    valorLedCocina = HIGH;

  }else if(valorLedCocina == HIGH && request.indexOf("/LED_COCINA=OFF") != -1){

    digitalWrite(pinLedCocina,LOW);
    valorLedCocina = LOW;

  }
```

Listing 6.19: Condición Para Activar/Desactivar Led Cocina

Por tanto, se observa que, para que se cumpla la condición de la cláusula if, el Led, en este caso, de la cocina, ha de estar apagado, es decir, la variable ha de tener un valor LOW, además, es necesario, que, la petición del cliente, asignada a la variable request de tipo String, contenga el conjunto de carácteres LED_COCINA=ON, permitiendo esto, ejecutar el código necesario para encender dicho LED.

Si no se cumplen estas condiciones, y, se cumple que, el LED esté encendido, y, en la petición del cliente, esté presente el String LED_COCINA=OFF, se cumplen las condiciones para ejecutar el código para apagar el LED.

Visto este fragmento de código, el mismo se repite para todos los LED's que se utilicen en la maqueta, cambiando las variables correspondientes. Además, se añade otra condición para poder controlar todas las luces al mismo tiempo, es decir, encender o apagar todas las luces al mismo tiempo.

```
/*Si se pide desde la web, encender todas las luces o apagarlos, se encienden o se
    apagan.*/
  if(request.indexOf("/LEDS=ON") != -1){

    digitalWrite(pinLedCocina,HIGH);
    valorLedCocina = HIGH;

```

```
    digitalWrite(pinLedEntrada,HIGH);
    valorLedEntrada = HIGH;

  }else if(request.indexOf("/LEDS=OFF") != -1){

    digitalWrite(pinLedCocina,LOW);
    valorLedCocina = LOW;

    digitalWrite(pinLedEntrada,LOW);
    valorLedEntrada = LOW;

  }
```

Listing 6.20: Control Maestro LED's

El mismo proceso se ha de seguir para el LED indicador del estado del aire acondicionado. Por lo que respecta a las peticiones sobre los servomotores que se encargan de subir o bajar las persianas, o de, abrir o cerrar las puertas del garaje o la puerta principal de la maqueta, el tratamiento de la petición del cliente, sería el siguiente.

```
/*Si existe un objeto delante de la puerta del garaje, o, desde la web de gestion
    se activa la puerta del garaje
 se realiza una espera de 2 segundos, y se acciona el servomotor que controla la
   puerta del garaje.

 Si se pide desde la web que se cierre la puerta, se esperan 2 segundos y se
   acciona el servomotor.*/

 if(cm <= 5 ||  request.indexOf("/GARAJE=ON") != -1){
   delay(2000);

   posicionServoGaraje = 90;
   servoGaraje.write(posicionServoGaraje);    //Servo en posicion 90 Grados (
   Abierta)
 }else  if(request.indexOf("/GARAJE=OFF") != -1){
   delay(2000);

   posicionServoGaraje = 0;
   servoGaraje.write(posicionServoGaraje);     //Servo en posicion 0 Grados (
   Cerrada)
 }

```

```
18  /*Si el sensor magnetico esta cerrado y se pide abrir la puerta principal desde
      la web de gestion, se acciona el
19  servomotor que controla la puerta.
20
21  Si el sensor magnetico esta abierto y se pide cerrar la puerta desde la web, se
      acciona el servomotor para cerrar
22  la puerta.*/
23
24  if(puertaPrincipal == 1 && request.indexOf("/ENTRADA=ON") != -1){
25
26    posicionServoEntrada = 90;
27    servoEntrada.write(posicionServoEntrada);    //Servo en posicion 90 Grados (
      Abierta)
28
29    puertaPrincipal = 0;     //Puerta Abierta
30
31  }else  if(puertaPrincipal == 0 && request.indexOf("/ENTRADA=OFF") != -1){
32
33    posicionServoEntrada = 0;
34    servoEntrada.write(posicionServoEntrada);     //Servo en posicion 0 Grados (
      Cerrada)
35
36    puertaPrincipal = 1;     //Puerta Cerrada
37
38  }
```

Listing 6.21: Control Servomotores Puerta Principal y Puerta Garaje

Para el control de los servomotores, se han de utilizar, tanto el valor entregado por el sensor de distancia colocado en la puerta del garaje, el cual nos da a conocer si existe un objeto o un vehículo cerca de dicha puerta, además de las variables de control de la posición del servomotor que ha de mover la puerta del garaje, la cual se ha declarado en el preámbulo del "sketch", asignando a la misma, la posición deseada en la que se ha de detener el servomotor, además, una vez asignada esta posición, mediante la llamada a la función write(), con argumento, la variable contenedora de la posición, se activa el servomotor, el cual se detendrá al alcanzar la posición de 90 grados, para, en este caso abrir la puerta del garaje, o la posición de 0 grados para cerrar la misma. El proceso para controlar las peticiones de control de los demás servomotores es muy similar a este, tan solo, se ha de cambiar el valor del grado dónde se desea que se detenga el servomotor.

En el caso del control de la puerta principal de entrada, se ha de tener en cuenta el valor del

sensor magnético conectado a la misma, para conocer si la puerta está abierta o cerrada, actuando en consecuencia, al abrir o cerrar la misma, ya sea, de forma física, cambiando el valor del sensor al abrir la puerta, o de forma remota, abriendo la puerta desde la página web de gestión.

Una vez visto el tratamiento de las peticiones del cliente, se puede pasar a analizar el código HTML necesario para crear una página de respuesta para las peticiones de los clientes.

```
/*CODIGO HTML DE LA PAGINA WEB DE RESPUESTA*/

  //Devuelve la respuesta
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println("Refresh: 60"); //Refrescar la pagina cada minuto
  client.println(""); //No olvidar la linea en blanco
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");
  client.println("<head>");
  client.println("<h1>Servidor Web</h1>");
  client.println("</head>");
  client.println("<body>");

  client.print("<div style=\"width:100%; height:100%;\">");
  client.print("<iframe src=\"https://www.zeitverschiebung.net/clock-widget-iframe
    -v2? language=es&timezone=Europe%2FMadrid\" width=\"100%\" height=\"150\"></
    iframe>");

  client.println("<h2>Informacion de la Casa</h2>");
  client.print("<b>Temperatura: </b>");
  client.print((long)celsius);
  client.print((char)176);  //Printear simbolo grado
  client.print("C");

  client.print("<hr style=\"border:15px,width:100%;\">");
  client.print("<br>");

  client.print("<br>");
  client.print("<br>");
```

Listing 6.22: Página de Respuesta del Servidor

Para poder programar código HTML sobre el servidor web alojado en la placa de desarrollo **NodeMCU**, se ha de hacer servir las llamadas a la función print() o println() que sean necesarias,

conteniendo estas, el código HTML en si de la página web de respuesta que lanza el servidor al atender las peticiones de los clientes. Así pues, en este fragmento de código, se crea la cabecera de la página, además de insertar un "iframe", que permita utilizar un reloj en tiempo real en la página web, además, se muestra también el valor de temperatura que se recibe del sensor, ya que, se imprime el valor de la variable "celsius", que, es la variable que contiene el valor que se recibe del sensor.

Siguiendo el código, en el siguiente fragmento, paso a analizar el código que permite controlar las luces LED conectadas a **NodeMCU**.

```
client.print("<div id=\"Luces\" style=\"text-align:center; width:100%; height:200
    px;\"> ");
  client.print("<h2 style=\"text-decoration:underline;\">Luces</h2>");

  /*CONTROL LUCES COMEDOR/COCINA*/
  client.print("<div id=\"Luces Comedor\" style=\"width:275px; height:100px;
    margin-left:200px; display:inline-block;\">");
  client.print("<b>Estado Luces Comedor/Cocina:</b> ");

  if(valorLedCocina == HIGH){
    client.print("<span style=\"color:green;\"><b>ON </b></span>");
  }else{
    client.print("<span style=\"color:red;\"><b>OFF </b></span>");
  }

  client.print("<br><br>");
  client.print("<a href=\"/LED_COCINA=ON\"\"><button style=\"background-color:
    green;\"><b>On</b> </button></a>");
  client.print("<a href=\"/LED_COCINA=OFF\"\"><button style=\"background-color:red
    ;\"><b>Off</b> </button></a>");
  client.print("</div>");
```

Listing 6.23: Control Luces Comedor/Cocina

Observando el código, se ve que, se crea un contenedor para albergar los diferentes botones y texto que conforman el elemento de control de las luces LED, dicho contenedor se crea a través de la etiqueta de HTML, para, dentro de este, crear otro contenedor que albergue el texto y los botones para encender o apagar los LED's de la cocina y el comedor. Así pues, para que, el cliente, al clickar en el botón de encendido o apagado, se encienda o apague el LED correspondiente, se utiliza la condición if, tal que, si la variable contenedora del valor de dicho LED tiene un valor HIGH, es decir, dicho LED, está encendido, el texto que se muestra

en el botón de encendido, será de color verde, en este caso, el texto no es ni más ni menos que ON, en caso contrario, el texto OFF del botón de apagado estará escrito en color rojo. Para la creación de los botones de control, como se puede ver, se utiliza la etiqueta <button>de HTML, además de, utilizar la etiqueta <a href>para que ejecute un link que cree una nueva página al clickar sobre el botón de encendido o apagado, ya que, es de esta manera, como el cliente envía peticiones al servidor, en este caso concreto, mediante el hipervínculo, se envía el argumento LED_COCINA=ON al servidor, el cual, como se ha visto en los anteriores fragmentos de código, lee la petición del cliente en busca de estos argumentos, y dependiendo de los mismos ejecuta diferentes acciones.

Visto este ejemplo, se deduce que, para el control de todos los elementos de la maqueta, se crean los respectivos contenedores, así como, diferentes botones para encender/apagar o abrir/cerrar los elementos de la maqueta.

Al finalizar el "sketch", el cliente se desconecta, y se vuelve a ejecutar el bucle de manera infinita.

```
1   client.println("</body>");
2   client.println("</html>");
3
4   delay(1);
5   Serial.println("Client Disconnected");
6   Serial.println("");
7
8 }
```

Listing 6.24: Terminación del Cliente

Finalizada esta sección, ya se conocen las características principales del "sketch" programado sobre **NodeMCU**.

7 Código Fuente Completo

Este apartado va a contener el código fuente completo que se cargará en la placa de desarrollo **NodeMCU**, habiendo sido explicadas ya, sus partes más importantes.

```
/*Librerias*/
#include <ESP8266WiFi.h>
#include <Servo.h>

/*Asignacion de pines de Arduino a los pines GPIO de NodeMCU LoLin*/
#define A0 A0        //Sensor Temperatura

#define D0 16        //Luces Entrada
#define D1 5         //Luces Cocina
#define D2 4         //Luz Aire Acondicionado
#define D3 0         //Trigger Sensor Distancia Garaje
#define D10 1        //Echo Sensor Distancia Garaje
#define D4 2         //Motor Puerta Entrada
#define D5 14        //Motor Ventilador
#define D6 12        //Motor Persiana Abajo
#define D7 13        //Motor Persiana Arriba
#define D8 15        //Sensor Magnetico Puerta Principal
#define D9 3         //Motor Puerta Garaje



//Conectar NodeMCU a red Wi-Fi
const char* ssid = "carricondo monter";
const char* password = "$carricondo2006$";


//Declaracion Pines Luces
int pinLedEntrada = 16;                          //GPIO 16 Led Entrada
int pinLedCocina = 5;                            //GPIO 05 Led Cocina Comedor
```

```
31
32
33 //Declaracion Pin Aire Acondicionado
34 int pinLedAC = 4;                                    //GPIO 04 Led Habitacion
35
36 //Declaracion Pines Motores
37 int pinMotorGaraje = 3;                              //GPIO 00 Motor Cochera
38 int pinMotorEntrada = 2;                             //GPIO 02 Motor Puerta Entrada
39 int pinMotorVentilador = 14;                         //GPIO 14 Motor Ventilador
40 int pinMotorPersianasAbajo = 12;                     //GPIO 12 Motor Persianas Abajo
41 int pinMotorPersianasArriba = 13;                    //GPIO 13 Motor Persianas
       Arriba
42
43
44 //Declaracion Pines Sensores
45 int pinSensorPuertaPrincipal = 15;                   //GPIO 15 Sensor Magnetico
       Puerta Principal
46 int triggerPinSensorCochera = 0;                     //GPIO 03 Trigger Sensor
       Distancia Puerta Cochera
47 int echoPinSensorCochera = 1;                       //GPIO 01 Echo Sensor Distancia
        Puerta Cochera
48 int pinSensorTemperatura = A0;                      //A0 Sensor Temperatura
49
50
51
52 /*DECLARACION OBJETOS */
53 Servo servoGaraje;
54 Servo servoEntrada;
55 Servo servoPersianaPisoAbajo;
56 Servo servoPersianaPisoArriba;
57 Servo servoVentilador;
58
59
60 int posicionServoGaraje;
61 int posicionServoEntrada;
62 int posicionServoPersianaPisoAbajo;
63 int posicionServoPersianaPisoArriba;
64 int posicionServoVentilador;
65
66
67 //Declaracion del puerto 80 como puerto del servidor
```

```
68 WiFiServer server(80);
69
70
71 void setup(){
72
73  Serial.begin(115200);
74  delay(10);
75
76  servoGaraje.attach(pinMotorGaraje);
77  servoEntrada.attach(pinMotorEntrada);
78  servoPersianaPisoAbajo.attach(pinMotorPersianasAbajo);
79  servoPersianaPisoArriba.attach(pinMotorPersianasArriba);
80  servoVentilador.attach(pinMotorVentilador);
81
82  posicionServoGaraje = 0;
83  posicionServoEntrada = 0;
84  posicionServoPersianaPisoAbajo = 0;
85  posicionServoPersianaPisoArriba = 0;
86  posicionServoVentilador = 90;     //Detenido
87
88  servoGaraje.write(posicionServoGaraje);
89  servoEntrada.write(posicionServoEntrada);
90  servoPersianaPisoAbajo.write(posicionServoPersianaPisoAbajo);
91  servoPersianaPisoArriba.write(posicionServoPersianaPisoArriba);
92  servoVentilador.write(posicionServoVentilador);
93
94  /*PinMode del sensor de TEMPERATURA*/
95  pinMode(pinSensorTemperatura, INPUT);
96
97  /*PinMode del sensor MAGNETICO de la PUERTA PRINCIPAL*/
98  pinMode(pinSensorPuertaPrincipal, INPUT);
99
100 /*PinMode del sensor de DISTANCIA de la puerta GARAJE*/
101 pinMode(triggerPinSensorCochera, OUTPUT);
102 pinMode(echoPinSensorCochera, INPUT);
103
104 /*PinMode de los LED*/
105 pinMode(pinLedEntrada, OUTPUT);
106 pinMode(pinLedCocina, OUTPUT);
107
108 digitalWrite(pinLedEntrada,LOW);
```

```
109  digitalWrite(pinLedCocina,LOW);
110
111  /*PinMode del Led del Aire Acondicionado*/
112  pinMode(pinLedAC, OUTPUT);
113  digitalWrite(pinLedAC,LOW);
114
115  //Conectar al WiFi
116  Serial.println();
117  Serial.println();
118  Serial.print("Conectando a ");
119  Serial.println(ssid);
120
121  WiFi.begin(ssid,password);
122
123  while(WiFi.status() != WL_CONNECTED){
124   delay(500);
125   Serial.print(".");
126  }
127
128  Serial.println("");
129  Serial.println("WiFi Connected");
130
131  //Iniciar el Servidor
132  server.begin();
133  Serial.println("Servidor Iniciado");
134
135  //Printear la direccion IP
136  Serial.print("Usa esta URL para conectarte: ");
137  Serial.print("http://");
138  Serial.print(WiFi.localIP());
139  Serial.println("/");
140
141 }//Final setup
142
143
144 void loop(){
145
146   /*LECTURA DE PINES DE ENTRADA/SALIDA*/
147
148   /*Lectura Pines LED*/
149   int valorLedCocina = digitalRead(pinLedCocina);
```

```
150  int valorLedEntrada = digitalRead(pinLedEntrada);
151
152
153  /*Lectura Pin Aire Acondicionado*/
154  int valorLedAC = digitalRead(pinLedAC);
155
156
157  /*Lectura Pin Termometro*/
158  float lectura = analogRead(pinSensorTemperatura);
159  //float millivolts = (lectura/1023.0)*5000;
160  float celsius = (3.3* lectura *100.0)/1024;
161  delay(1000);
162
163  /*Lectura Sensor Magnetico Puerta Principal*/
164  int puertaPrincipal = digitalRead(pinSensorPuertaPrincipal);
165
166  /*Lectura Sensor Distancia Puerta Garaje (Mediante Funcion Auxiliar)*/
167  int cm = ping(triggerPinSensorCochera,echoPinSensorCochera);
168
169  /*
      ----------------------------------------------------------------------------------------------------
     */
170
171  /*CODIGO DEL SERVIDOR WEB*/
172
173
174  //Comprobar si hay algun cliente conectado
175  WiFiClient client = server.available();
176
177  if(!client){
178    return;
179  }
180
181  //Esperar hasta que el cliente envie algun dato
182  Serial.println("new client");
183  while(!client.available()){
184    delay(1);
185  }
186
187
188  //Leer la primera linea de la peticion
```

```
189  String request = client.readStringUntil('/r');
190  Serial.println(request);
191  client.flush();
192
193  /*---------------------------------------------------- GET LUCES
       ----------------------------------------------------*/
194
195  /*Si el led de la cocina o entrada, estan apagados, y, desde la web se pide
       encenderlos, se encienden.
196
197  Si estan encendidos y se pide apagarlos desde la web, se apagan.*/
198
199  if(valorLedCocina == LOW && request.indexOf("/LED_COCINA=ON") != -1){
200
201    digitalWrite(pinLedCocina,HIGH);
202    valorLedCocina = HIGH;
203
204  }else if(valorLedCocina == HIGH && request.indexOf("/LED_COCINA=OFF") != -1){
205
206    digitalWrite(pinLedCocina,LOW);
207    valorLedCocina = LOW;
208
209  }
210
211  if(valorLedEntrada == LOW && request.indexOf("/LED_ENTRADA=ON") != -1){
212
213    digitalWrite(pinLedEntrada,HIGH);
214    valorLedEntrada = HIGH;
215
216  }else if(valorLedEntrada == HIGH && request.indexOf("/LED_ENTRADA=OFF") != -1){
217
218    digitalWrite(pinLedEntrada,LOW);
219    valorLedEntrada = LOW;
220
221  }
222
223
224  /*Si se pide desde la web, encender todas las luces o apagarlos, se encienden o
       se apagan.*/
225  if(request.indexOf("/LEDS=ON") != -1){
226
```

```
227     digitalWrite(pinLedCocina,HIGH);
228     valorLedCocina = HIGH;
229
230     digitalWrite(pinLedEntrada,HIGH);
231     valorLedEntrada = HIGH;
232
233     //digitalWrite(pinLedPatio,HIGH);
234     //valorLedPatio = HIGH;
235
236   }else if(request.indexOf("/LEDS=OFF") != -1){
237
238     digitalWrite(pinLedCocina,LOW);
239     valorLedCocina = LOW;
240
241     digitalWrite(pinLedEntrada,LOW);
242     valorLedEntrada = LOW;
243
244     //digitalWrite(pinLedPatio,LOW);
245     //valorLedPatio = LOW;
246
247   }
248
249   /*------------------------------------------------------ GET AC
        ------------------------------------------------------*/
250
251   /*Si el led del AC esta apagado, y se pide encenderlo desde la web, o, la
        temperatura registrada
252   por el sensor, supera los 26 grados centigrados, se enciendo el led que simula
        el AC.
253
254   Si el led esta encendido, y, se pide apagarlo o la temperatura baja de 26 grados
        se apaga el led
255   simulador del AC*/
256
257   if(valorLedAC == LOW && request.indexOf("/LED_AC=ON") != -1 || celsius >= 26){
258
259     digitalWrite(pinLedAC,HIGH);
260     valorLedAC = HIGH;
261
262   }else if(valorLedAC == HIGH && request.indexOf("/LED_AC=OFF") != -1 || celsius
        <= 26){
```

```
263
264     digitalWrite(pinLedAC,LOW);
265     valorLedAC = LOW;
266
267   }
268
269
270   /*---------------------------------------------------------- GET PUERTAS
        ----------------------------------------------------------*/
271
272   /*Si existe un objeto delante de la puerta del garaje, o, desde la web de
        gestion se activa la puerta del garaje
273   se realiza una espera de 2 segundos, y se acciona el servomotor que controla la
        puerta del garaje.
274
275   Si se pide desde la web que se cierre la puerta, se esperan 2 segundos y se
        acciona el servomotor.*/
276
277   if(cm <= 5 ||   request.indexOf("/GARAJE=ON") != -1){
278     delay(2000);
279
280     posicionServoGaraje = 90;
281     servoGaraje.write(posicionServoGaraje);    //Servo en posicion 90 Grados (
        Abierta)
282   }else   if(request.indexOf("/GARAJE=OFF") != -1){
283     delay(2000);
284
285     posicionServoGaraje = 0;
286     servoGaraje.write(posicionServoGaraje);     //Servo en posicion 0 Grados (
        Cerrada)
287   }
288
289   /*Si el sensor magnetico esta cerrado y se pide abrir la puerta principal desde
        la web de gestion, se acciona el
290   servomotor que controla la puerta.
291
292   Si el sensor magnetico esta abierto y se pide cerrar la puerta desde la web, se
        acciona el servomotor para cerrar
293   la puerta.*/
294
295   if(puertaPrincipal == 1 && request.indexOf("/ENTRADA=ON") != -1){
```

```
296
297     posicionServoEntrada = 90;
298     servoEntrada.write(posicionServoEntrada);    //Servo en posicion 90 Grados (
        Abierta)
299
300     puertaPrincipal = 0;     //Puerta Abierta
301
302   }else   if(puertaPrincipal == 0 && request.indexOf("/ENTRADA=OFF") != -1){
303
304     posicionServoEntrada = 0;
305     servoEntrada.write(posicionServoEntrada);     //Servo en posicion 0 Grados (
        Cerrada)
306
307     puertaPrincipal = 1;     //Puerta Cerrada
308
309   }
310
311
312   /*------------------------------------------------------ GET PERSIANAS
        ------------------------------------------------------*/
313
314   /*Si se pide desde web subir la persiana del piso de abajo, se acciona el
        servomotor.
315
316   Si se pide bajarla, se acciona el servomotor hasta llegar a la posicion inicial.
        */
317   if(request.indexOf("/PERSIANA_ABAJO=ON") != -1){
318
319     posicionServoPersianaPisoAbajo = 90;
320     servoPersianaPisoAbajo.write(posicionServoPersianaPisoAbajo);    //Servo en
        posicion 90 Grados (Abierta)
321
322   }else if(request.indexOf("/PERSIANA_ABAJO=OFF") != -1){
323
324     posicionServoPersianaPisoAbajo = 0;
325     servoPersianaPisoAbajo.write(posicionServoPersianaPisoAbajo);     //Servo en
        posicion 0 Grados (Cerrada)
326
327   }
328
329
```

```
330
331 if(request.indexOf("/PERSIANA_ARRIBA=ON") != -1){
332
333   posicionServoPersianaPisoArriba = 90;
334   servoPersianaPisoArriba.write(posicionServoPersianaPisoArriba);    //Servo en
      posicion 90 Grados (Abierta)
335
336 }else  if(request.indexOf("/PERSIANA_ARRIBA=OFF") != -1){
337
338   posicionServoPersianaPisoArriba = 0;
339   servoPersianaPisoArriba.write(posicionServoPersianaPisoArriba);     //Servo en
      posicion 0 Grados (Cerrada)
340
341 }
342
343
344 /*------------------------------------------------------ GET VENTILADOR
      ------------------------------------------------------*/
345
346 /*Si se pide accionar el ventilador desde la web, se acciona el servomotor que
      lo controla.*/
347 if(request.indexOf("/VENTILADOR=ON") != -1){
348
349   posicionServoVentilador = 0;     //Horario(Movimiento)
350   servoVentilador.write(posicionServoVentilador);    //Servo en rotacion en
      sentido horario (Movimiento)
351
352 }else if(request.indexOf("/VENTILADOR=OFF") != -1){
353
354   posicionServoVentilador = 90;  //Detenido
355   servoVentilador.write(posicionServoVentilador);    //Servo detenido
356
357 }
358
359
360 /*CODIGO HTML DE LA PAGINA WEB DE RESPUESTA*/
361
362
363 //Devuelve la respuesta
364 client.println("HTTP/1.1 200 OK");
365 client.println("Content-Type: text/html");
```

```
366  client.println("Refresh: 60"); //Refrescar la pagina cada minuto
367  client.println(""); //No olvidar la linea en blanco
368  client.println("<!DOCTYPE HTML>");
369  client.println("<html>");
370  client.println("<head>");
371  client.println("<h1>Servidor Web</h1>");
372  client.println("</head>");
373  client.println("<body>");
374
375  client.print("<div style=\"width:100%; height:100%;\">");
376  client.print("<iframe src=\"https://www.zeitverschiebung.net/clock-widget-iframe
      -v2? language=es&timezone=Europe%2FMadrid\" width=\"100%\" height=\"150\"></
      iframe>");
377
378  client.println("<h2>Informacion de la Casa</h2>");
379  client.print("<b>Temperatura: </b>");
380  client.print((long)celsius);
381  client.print((char)176);  //Printear simbolo grado
382  client.print("C");
383
384  client.print("<hr style=\"border:15px,width:100%;\">");
385  client.print("<br>");
386
387
388  client.print("<br>");
389  client.print("<br>");
390
391  /*
      ----------------------------------------------------------------------------
      */
392
393  client.print("<div id=\"Luces\" style=\"text-align:center; width:100%; height
      :200px;\"> ");
394  client.print("<h2 style=\"text-decoration:underline;\">Luces</h2>");
395
396  /*CONTROL LUCES COMEDOR/COCINA*/
397  client.print("<div id=\"Luces Comedor\" style=\"width:275px; height:100px;
      margin-left:200px; display:inline-block;\">");
398  client.print("<b>Estado Luces Comedor/Cocina:</b> ");
399
400  if(valorLedCocina == HIGH){
```

```
401     client.print("<span style=\"color:green;\"><b>ON </b></span>");
402   }else{
403     client.print("<span style=\"color:red;\"><b>OFF </b></span>");
404   }
405
406   client.print("<br><br>");
407   client.print("<a href=\"/LED_COCINA=ON\"\"><button style=\"background-color:
        green;\"><b>On</b> </button></a>");
408   client.print("<a href=\"/LED_COCINA=OFF\"\"><button style=\"background-color:red
        ;\"><b>Off</b> </button></a>");
409   client.print("</div>");
410
411
412   /*CONTROL LUCES ENTRADA*/
413   client.print("<div id=\"Luces Entrada\" style=\"width:250px; height:100px;
        display:inline-block;\">");
414   client.print("<b>Estado Luces Entrada:</b> ");
415
416   if(valorLedEntrada == HIGH){
417     client.print("<span style=\"color:green;\"><b>ON </b></span>");
418   }else{
419     client.print("<span style=\"color:red;\"><b>OFF </b></span>");
420   }
421
422   client.print("<br><br>");
423   client.print("<a href=\"/LED_ENTRADA=ON\"\"><button style=\"background-color:
        green;\"><b>On</b> </button></a>");
424   client.print("<a href=\"/LED_ENTRADA=OFF\"\"><button style=\"background-color:
        red;\"><b>Off</b> </button></a>");
425   client.print("</div>");
426
427
428    /*CONTROL TOTAL LUCES*/
429   client.print("<div id=\"Control Luces\" style=\"float:right; width:175px; height
        :100px; display:inline-block; margin-right:25px;\">");
430   client.print("<b>Control Maestro de Luces</b>");
431
432   client.print("<br><br>");
433   client.print("<a href=\"/LEDS=ON\"\"><button style=\"background-color:green;\"><
        b>On</b> </button></a>");
```

```
434   client.print("<a href=\"/LEDS=OFF\"\"><button style=\"background-color:red;\"><b
        >Off</b> </button></a>");
435   client.print("</div>");
436   client.print("</div>");
437
438
439 /*
        ---------------------------------------------------------------------------
        */
440
441   client.print("<div id=\"Puertas\" style=\"text-align:center; width:100%; height
        :200px; \"> ");
442   client.print("<h2 style=\"text-decoration:underline;\">Puertas</h2>");
443
444   /*CONTROL PUERTA GARAJE*/
445   client.print("<div id=\"Puerta Garaje\" style=\"width:250px; height:100px;
        display:inline-block;\">");
446   client.print("<b>Estado Puerta Garaje:</b> ");
447
448   if(posicionServoGaraje == 90){
449     client.print("<span style=\"color:green;\"><b>Abierta </b></span>");
450   }else if(posicionServoGaraje == 0){
451     client.print("<span style=\"color:red;\"><b>Cerrada </b></span>");
452   }
453
454   client.print("<br><br>");
455   client.print("<a href=\"/GARAJE=ON\"\"><button style=\"background-color:green
        ;\"><b>Abrir </b> </button></a>");
456   client.print("<a href=\"/GARAJE=OFF\"\"><button style=\"background-color:red
        ;\"><b>Cerrar </b> </button></a>");
457   client.print("</div>");
458
459
460
461   /*CONTROL PUERTA ENTRADA*/
462   client.print("<div id=\"Puerta Entrada\" style=\"width:250px; height:100px;
        display:inline-block;\">");
463   client.print("<b>Estado Puerta Entrada:</b> ");
464
465   if(puertaPrincipal == 0){
466     client.print("<span style=\"color:green;\"><b>Abierta </b></span>");
```

```
467   }else if(puertaPrincipal == 1){
468     client.print("<span style=\"color:red;\"><b>Cerrada </b></span>");
469   }
470
471   client.print("<br><br>");
472   client.print("<a href=\"/ENTRADA=ON\"\"><button style=\"background-color:green
        ;\"><b>Abrir </b> </button></a>");
473   client.print("<a href=\"/ENTRADA=OFF\"\"><button style=\"background-color:red
        ;\"><b>Cerrar </b> </button></a>");
474   client.print("</div>");
475   client.print("</div>");
476
477
478   /*
        ----------------------------------------------------------------------------
        */
479
480   client.print("<div id=\"Persianas\" style=\"text-align:center; width:100%;
        height:200px;\"> ");
481   client.print("<h2 style=\"text-decoration:underline;\">Persianas</h2>");
482
483   /*CONTROL PERSIANAS COMEDOR/COCINA*/
484   client.print("<div id=\"Persiana Comedor\" style=\"width:325px; height:100px;
        display:inline-block;\">");
485   client.print("<b>Estado Persianas Comedor/Cocina:</b> ");
486
487   if(posicionServoPersianaPisoAbajo == 90){
488     client.print("<span style=\"color:green;\"><b>Abierta </b></span>");
489   }else if(posicionServoPersianaPisoAbajo == 0){
490     client.print("<span style=\"color:red;\"><b>Cerrada </b></span>");
491   }
492
493   client.print("<br><br>");
494   client.print("<a href=\"/PERSIANA_ABAJO=ON\"\"><button style=\"background-color:
        green;\"><b>Abrir </b> </button></a>");
495   client.print("<a href=\"/PERSIANA_ABAJO=OFF\"\"><button style=\"background-color
        :red;\"><b>Cerrar </b> </button></a>");
496   client.print("</div>");
497
498   /*CONTROL PERSIANAS HABITACION*/
```

```
499   client.print("<div id=\"Persiana Habitacion\" style=\"width:300px; height:100px;
        display:inline-block;\">");
500   client.print("<b>Estado Persianas Habitacion:</b> ");
501
502   if(posicionServoPersianaPisoArriba == 90){
503     client.print("<span style=\"color:green;\"><b>Abierta </b></span>");
504   }else if(posicionServoPersianaPisoArriba == 0){
505     client.print("<span style=\"color:red;\"><b>Cerrada </b></span>");
506   }
507
508   client.print("<br><br>");
509   client.print("<a href=\"/PERSIANA_ARRIBA=ON\"\"><button style=\"background-color
        :green;\"><b>Abrir </b> </button></a>");
510   client.print("<a href=\"/PERSIANA_ARRIBA=OFF\"\"><button style=\"background-
        color:red;\"><b>Cerrar </b> </button></a>");
511   client.print("</div>");
512   client.print("</div>");
513
514
515
516   /*
        ---------------------------------------------------------------------------
        */
517
518   client.print("<div id=\"Aire_Acondicionado\" style=\"text-align:center; width
        :100%; height:200px;\"> ");
519   client.print("<h2 style=\"text-decoration:underline;\">Aire Acondicionado</h2>")
        ;
520
521   /*CONTROL A/C COMEDOR Y HABITACION*/
522   client.print("<div id=\"Aire Acondicionado\" style=\"width:400px; height:100px;
        display:inline-block;\">");
523   client.print("<b>Estado Aire Acondicionado Comedor/Cocina:</b> ");
524
525   if(valorLedAC == HIGH){
526     client.print("<span style=\"color:green;\"><b>ON </b></span>");
527   }else{
528     client.print("<span style=\"color:red;\"><b>OFF </b></span>");
529   }
530
531   client.print("<br><br>");
```

```
532  client.print("<a href=\"/LED_AC=ON\"\"><button style=\"background-color:green
       ;\"><b>On </b> </button></a>");
533  client.print("<a href=\"/LED_AC=OFF\"\"><button style=\"background-color:red
       ;\"><b>Off </b> </button></a>");
534  client.print("</div>");
535  client.print("</div>");
536
537
538  client.print("</div>");
539
540
541  /*
       ----------------------------------------------------------------------------
       */
542
543  client.print("<div id=\"Ventilador\" style=\"text-align:center; width:100%;
       height:200px;\"> ");
544  client.print("<h2 style=\"text-decoration:underline;\">Ventilador</h2>");
545
546  /*CONTROL VENTILADOR*/
547  client.print("<div id=\"Ventilador\" style=\"width:300px; height:100px; display:
       inline-block;\">");
548  client.print("<b>Estado Ventilador:</b> ");
549
550  if(posicionServoVentilador == 0){
551    client.print("<span style=\"color:green;\"><b>Movimiento </b></span>");
552  }else if(posicionServoVentilador == 90){
553    client.print("<span style=\"color:red;\"><b>Detenido </b></span>");
554  }
555
556  client.print("<br><br>");
557  client.print("<a href=\"/VENTILADOR=ON\"\"><button style=\"background-color:
       green;\"><b>Abrir </b> </button></a>");
558  client.print("<a href=\"/VENTILADOR=OFF\"\"><button style=\"background-color:red
       ;\"><b>Cerrar </b> </button></a>");
559  client.print("</div>");
560
561  /*
       ----------------------------------------------------------------------------
       */
562
```

```
563   client.println("</body>");
564   client.println("</html>");
565
566   delay(1);
567   Serial.println("Client Disconnected");
568   Serial.println("");
569
570 }
571
572
573 /*Funcion Auxiliar Sensor Distancia*/
574
575 int ping(int triggerPinSensorCochera, int echoPinSensorCochera){
576
577   long duration,distanceCm;
578
579   digitalWrite(triggerPinSensorCochera, LOW); //Generacion de un pulso limpio
580   delayMicroseconds(4);
581   digitalWrite(triggerPinSensorCochera, HIGH); //Generamos senal
582   delayMicroseconds(10);
583   digitalWrite(triggerPinSensorCochera,LOW);
584
585   duration = pulseIn(echoPinSensorCochera, HIGH); //Medimos el tiempo entre pulsos
586
587   distanceCm = duration * 10/ 292 /2; //Convertimos distancia a cm
588   return distanceCm;
589
590 }
```

Listing 7.1: Código Fuente Completo del Proyecto

8 Fucionamiento de la Página Web de Gestión

En esta parte de la memoria, voy a ilustrar, mediante diferentes fotografías, el panel de control que ha de manejar la mayoría de los elementos del sistema, es decir, la apariencia que tiene la página web de respuesta que ha de mostrar el servidor web.

Dicho esto, la cabecera de la página, ha de mostrar información básica del sistema, como puede ser, la fecha y hora actual, además, del valor de la temperatura que detecta el sensor.

Figura 25: Cabecera Página Web

Justo después de la cabecera, se empiezan a colocar los botones y elementos necesarios para el control de los actuadores del sistema, en este caso, y, en primer lugar, se han de controlar las luces LED del sistema, para esto, decido colocar cuatro elementos , cada uno, contenedor de, las luces LED concretas de una estancia, y, el último, un controlador maestro de todas las luces conectadas al sistema.

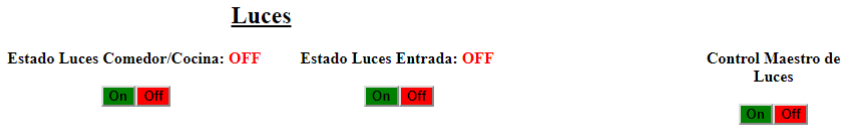


Figura 26: Control Luces

Se observa, además, que, al estar una luz encendida, o encenderla, mediante el botón habilitado para ello, el estado mostrado en la página web del sistema, pasa de OFF a ON, como se muestra en la siguiente captura de pantalla.

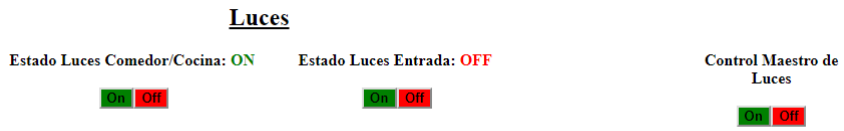


Figura 27: Luz Encendida

De la misma manera, siguiendo con la composición de la página web de gestión del sistema, justo debajo del apartado de luces, se muestra el apartado de control de las puertas, siendo este, ante todo informativo del estado de las mismas, es decir, si las puertas se encuentran cerradas o abiertas, se muestra el texto correspondiente al estado, ya sea abierta o cerrada, siendo posible la apertura de las mismas si fuera necesario, mediante los botones habilitados para ello.

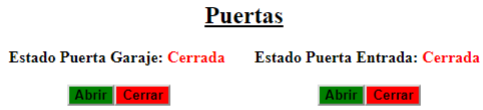


Figura 28: Gestión Puertas

Se muestra justo a continuación, los botones y texto necesarios para el control de las persianas de la maqueta, que, al igual que pasa con las puertas, se puede saber si están abiertas o cerradas, o abrirlas y cerrarlas desde la web.

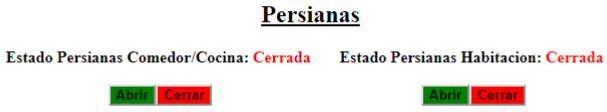


Figura 29: Gestión Persianas

Después, se encuentra el apartado para el control del aire acondicionado, el cual, puede ser encendido automáticamente, desde el "sketch", asignando una temperatura umbral a partir de la cual se deba encender el aire acondicionado, además de, la opción de encenderlo directamente desde la web de gestión del sistema, mediante los correspondientes botones de encendido y apagado.

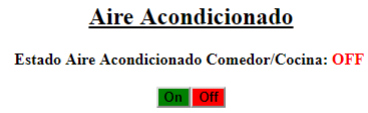


Figura 30: Gestión Aire Acondicionado

Por último, se encuentra la sección para el control del ventilador, y, al igual que en las secciones anteriores, se puede conocer el estado del mismo, es decir, si está en funcionamiento o detenido, además de, disponer de la opción de encenderlo o apagarlo desde la misma página web.

Figura 31: Gestión Ventilador

Finaliza así, la revisión de la página web creada para la gestión de este sistema domótico.

9 Mejoras o Proyectos con NodeMCU

En esta sección de la memoria, voy a intentar formular algunas posibles mejoras a realizar en este proyecto, así como, otros usos que puede tener la placa de desarrollo utilizada en este proyecto, la **NodeMCU** LoLin V3.

Una posible mejora para este proyecto, sería la de añadir más sensores y actuadores a la misma, por ejemplo, un lector de tarjetas NFC o RFID para controlar los accesos, sensores de inundación, vibración o de gases, para detectar anomalías naturales que pudieran aparecer en el entorno, tales como, inundaciones, terremotos o incendios, además de otro tipos de sensores. Aunque, el problema de esta mejora, es el hecho de que, la placa **NodeMCU** no dispone de suficientes pines de entrada/salida, hecho que provoca no poder aumentar el número de sensores y actuadores.

Una posible solución para este problema, sería, la inclusión de varias placas **NodeMCU** a las que conectar los diferentes sensores o actuadores, debiendo así, encontrar la manera de poder controlar todas las placas que pudieran conformar este sistema desde un mismo sitio, ya fuera, una web desde donde controlar los elementos o una aplicación móvil, además de encontrar la manera o el protocolo por el que se han de comunicar las diferentes placas, o nodos del sistema.

Otro posible proyecto con esta placa, sería, intentar utilizar esta placa con protocolos estandarizados del mundo de la domótica, tales como, ZigBee, con el cual se podría crear una red de nodos conectados de manera inalámbrica entre sí.

Además, se puede utilizar esta placa para controlar relés de diferentes voltajes, para, por ejemplo, crear un interruptor inalámbrico para controlar las luces de un hogar.

Existen también, diferentes usos para esta placa de desarrollo, aunque, algo más cercanos al mundo del hacking y el cracking, ya que, sobre todo, en los Estados Unidos, se usa esta placa para colapsar los receptores de redes Wi-Fi de los dispositivos móviles, o como se denomina en Inglés, Wi-Fi Jamming, el funcionamiento sería, gracias a las características de esta placa, y, el hecho de que, puede crear sus propios puntos de acceso, se programa esta placa para que cree infinidad de puntos de acceso para confundir y colapsar los posibles receptores de redes Wi-Fi que estén a su alrededor, en este caso, todo tipo de smartphones, tablets y otros dispositivos móviles.

10 Conclusiones

Como conclusión del proyecto, paso a ilustrar mis pensamientos sobre el mismo y sobre la placa utilizada.

En primer lugar, decidí utilizar esta placa de desarrollo por el gran futuro que creo que puede tener, ya que, básicamente, se trata de una placa **Arduino** con conectividad Wi-Fi, por un cuarto del precio que costaría dotar la placa **Arduino** más barata de este tipo de conectividad, abriendo así, un abanico de posibilidades y proyectos muy extenso, tan solo, hay que visitar el foro dedicado al SoC Wi-Fi del que hace uso esta placa, para darse cuenta de la cantidad de desarrolladores y proyectos que existen utilizando la misma.

Además, decidí realizar este proyecto debido a mis inquietudes personales con el mundo de la domótica y la inmótica, ya que, a mi parecer, será uno de los grandes mercados de la tecnología dentro de unos años, y, me parecía que, este proyecto, podía ser una buena aproximación o iniciación en este sector tecnológico.

Una vez terminado, puedo decir que me ha parecido un proyecto muy interesante y, del cual, he aprendido mucho sobre las nuevas tecnologías aplicadas a la domótica y la inmótica, así como, del Internet de las Cosas y de la filosofía del "Do It Yourself".

Bibliografía

[1] Foro No-Oficial del SoC ESP8266 http://www.esp8266.com/index.php?sid=2b69df7b69ed6ddfad316cf7ab34e6dc

[2] Pagina Web Oficial de NodeMCU http://nodemcu.com/index_en.html

[3] Documentacion NodeMCU https://nodemcu.readthedocs.io/en/master/

[4] Tutorial para programar NodeMCU desde IDE Arduino http://www.prometec.net/nodemcu-arduino-ide/

[5] Diferentes tutoriales para el control de sensores en Arduino https://www.luisllamas.es/

[6] Tutoriales para el control de servomotores en Arduino https://programarfacil.com/tutoriales/fragmentos/servomotor-con-arduino/

[7] Pagina Wikipedia sobre Open-Hardware https://es.wikipedia.org/wiki/Hardware_libre

[8] Pagina Wikipedia sobre Domotica https://es.wikipedia.org/wiki/Dom%C3%B3tica

[9] Pagina Web Oficial KNX https://www.knx.org/es/

[10] Pagina Web Oficial ZigBee http://www.zigbee.org/

[11] Pagina Wikipedia sobre X10 https://es.wikipedia.org/wiki/X10

[12] Pagina Web Oficial Z-Wave http://www.z-wave.com/

[13] Pagina Web Oficial EnOcean https://www.enocean.com/en/

[14] Pagina Wikipedia sobre Inmotica https://es.wikipedia.org/wiki/Inm%C3%B3tica

[15] Pagina Web Oficial Apple Homekit https://www.apple.com/es/shop/accessories/all-accessories/homekit

[16] Pagina Web Oficial Samsung Smart Home http://www.samsung.com/es/a-fondo/smart-home/

[17] Pagina Web Oficial Google Home https://madeby.google.com/home/

[18] Pagina Web Oficial Amazon Echo https://www.amazon.com/Amazon-Echo-Bluetooth-Speaker-with-WiFi-Alexa/dp/B00X4WHP5E

[19] Pagina Web Oficial Loxone https://www.loxone.com/eses/

[20] Pagina Web Oficial Arduino https://www.arduino.cc/

[21] Pagina Web Oficial Raspberry Pi https://www.raspberrypi.org/

[22] Pagina Web Oficial Souliss http://souliss.net/

[23] Pagina Web Oficial DomoticZ https://domoticz.com/

[24] Pagina Web Oficial PiHome http://pihome.harkemedia.de/

[25] Pagina Web Oficial Jeedom https://www.jeedom.com/site/fr/