Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

Red sensorial para la asistencia a personas dependientes

Autor: Alberto Ramos de la Torre

Tutor/es: Juan Jesús García Domínguez

Salvador Gómez Pedraz

2019

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

Grado en Ingeniería Electrónica y Automática Industrial

Trabajo Fin de Grado

Red sensorial para la asistencia a personas dependientes

Autor: Alberto Ramos de la Torre

Tutor: Juan Jesús García Domínguez/ Salvador Gómez Pedraz

TRIBUNAL:

Presidente: José Manuel Villadangos Carrizo

Vocal 1º: Ana Jiménez Martín

Vocal 2º: Juan Jesús García Domínguez

Calificación: ……..

Fecha: ……

RESUMEN

En la actualidad, con el avance de las nuevas tecnologías y el desarrollo de las redes inalámbricas, cada vez más objetos y dispositivos están conectados entre ellos y a su vez, a Internet. Esto se conoce como Internet de las cosas o Internet of things.

El objetivo de este proyecto será realizar el diseño de un sistema de varios clientes, Arduinos WiFi, con un servidor, un Arduino Ethernet, cuya información recibida será enviada a una página web donde se podrá evaluar. Para realizar el proyecto se necesitará diseñar y montar una serie de circuitos electrónicos además de programar correctamente el sistema para que realice las funciones que se demandan. Concretamente, este proyecto está enfocado a la asistencia de personas dependientes.

Palabras clave: IoT, Arduino Ethernet, Arduino WiFi, servidor, cliente.

ABSTRACT

Nowadays, with the advancement of new technologies and the development of wireless networks, more and more objects and devices are connected to each other and, in turn, to the Internet. This is known as Internet of things.

The purpose of this project will be to design a system with several clients, in this case, Arduinos WiFi, and a server, an Arduino Ethernet, whose received information will be sent to a web page where it can be evaluated. To carry out the project it will be necessary to design and assemble some electronic circuits. In addition, it will be necessary to correctly program the system to perform the demanded function. Specifically, this project is focused on the assistance of dependents.

Keywords: IoT, Arduino Ethernet, Arduino WiFi, Server, Client.

RESUMEN EXTENDIDO

Hasta hace poco tiempo era difícil pensar que se podría tener conexión a internet en un teléfono móvil o en un simple reloj de pulsera. Sin embargo, hoy en día, debido a que las tecnologías de la información y comunicación son cada vez más accesibles a la mayoría de la población, ha surgido el llamado Internet de las cosas.

El Internet de las cosas (IoT) integra una red de dispositivos físicos con electrónica, software, sensores, actuadores y conectividad que permite que objetos cotidianos se conecten, recopilen e intercambien datos. Eventualmente, los datos recopilados y la información fusionada se conectan a Internet, para brindar oportunidades para construir sistemas y aplicaciones inteligentes.

El nuevo estudio 'The Internet of Things: Today and Tomorrow' publicado por Aruba, empresa de Hewlett Packard Enterprise, revela que el Internet de las Cosas será adoptado de manera masiva muy pronto, ya que el 85% de las empresas planean implementarlo para 2019.

El IoT se ha comenzado a adoptar fuertemente en muchos sectores de la sociedad e industria como pueden ser los siguientes (Aruba, s.f.):

- El sector industrial aumenta la eficiencia y la visibilidad de los negocios gracias a la monitorización y el mantenimiento habilitados para el IoT.

- Los minoristas se involucran con los clientes y aumentan las ventas utilizando la tecnología de ubicación en interiores basadas en monitoreo y mantenimiento.

- Las empresas crean lugares de trabajo inteligentes para mejorar la productividad y la eficiencia.

- El sector salud (healthcare) implementa el IoT para mejorar la monitorización de pacientes, reducir costes y fomentar la innovación.

Precisamente este proyecto se va a enfocar en la implementación de esta tecnología en el sector de la salud, permitiendo a los usuarios comunicarse con los profesionales mediante el envío de una señal de emergencia. Además, los responsables de la supervisión del estado de las personas dependientes la recibirán en tiempo real, pudiendo, de esta forma, mandar una respuesta rápida si fuera necesario. Esta tecnología produciría una mejora en la respuesta ante situaciones de emergencia y una mejor atención del profesional hacia el paciente.

Para realizar este prototipo se diseñará todo el software y el hardware para la correcta comunicación entre dos clientes y un servidor, que mandará la información a una página web.

Contenido

TABLA DE ILUSTRACIONES

1. MOTIVACIÓN Y OBJETIVOS

La tecnología IoT se ha implementado en la mayoría de los ámbitos de la vida del ser humano y no sólo está de paso, sino que ha llegado para quedarse. Por lo tanto, es inevitable no sentir curiosidad acerca de esta tecnología y de las posibilidades que ofrecen los distintos dispositivos para desarrollarla en determinados proyectos.

Además, debido a que ya en prácticamente todos los lugares y edificios de las ciudades existe conexión WiFi a internet se ha pensado en la realización de este proyecto, donde se busca una comunicación optimizada y rápida entre clientes y servidor.

Más concretamente, me he centrado en la inclusión en el ámbito de la salud o del cuidado de personas dependientes debido a que en este sector es vital la rápida comunicación entre paciente y profesional y, además, la búsqueda de un método sencillo y barato del mismo que se pueda ampliar en el futuro añadiendo diversos módulos para su correcta monitorización y que los encargados tengan más información sobre los pacientes.

Para la realización de este prototipo se opta por la utilización de placas microcontroladoras Arduino WiFi en cada habitación al que se le conectarán pulsadores que se podrán activar y mandarán su señal a un microcontrolador Ethernet que se ubicará en cada planta del edificio. Todos los microcontroladores WiFi tendrán conexión con el micro Ethernet que actúa como servidor. Además, los profesionales de la residencia podrán acceder a un servidor web para recibir la información y poder generar una señal de vuelta que los pacientes podrán recibir mediante señales luminosas de bajo consumo diferenciadas por colores específicos en función del estado de atención de la solicitud.

El objetivo principal será el diseño y la correcta programación de la aplicación para que lleve a cabo las funciones que pretendemos.

Además, para la consecución del objetivo principal, se deben alcanzar los siguientes objetivos intermedios:

- Estudio de las tecnologías de comunicación inalámbricas que mejor se pueden adaptar al proyecto.

- Investigar la documentación y el funcionamiento de los microcontroladores y de la programación que se va a utilizar en el proyecto

- Diseño del Hardware y el posterior montaje de los circuitos electrónicos.

- Diseño del software y programar el código necesario que realice un correcto funcionamiento del sistema de la forma más eficiente posible. Además, realizar una página web donde se muestre la información recogida en los clientes.

- Realización de las pruebas comprobando que el sistema funciona correctamente y obtener las conclusiones finales.

- Mostrar conocimientos adquiridos en el grado cursado mediante la programación y el diseño de los circuitos electrónicos introducidos.

- Búsqueda de nuevas aplicaciones para el IoT así como posibles desarrollos de este mismo proyecto que puedan realizarse en este sector o en otros nuevos.

2. ESTADO DEL ARTE

2.1 EL INTERNET DE LAS COSAS (IoT)

Internet de las cosas es una red de objetos físicos –vehículos, máquinas, electrodomésticos y más– que utiliza sensores y aplicaciones para conectarse e intercambiar datos por internet. Básicamente se basa en la interconexión de cualquier producto con otro de su alrededor. El objetivo es que todos los dispositivos se comuniquen entre sí y así ser más inteligentes e independientes.

Ilustración 1: internet de las cosas

Este término fue adoptado a finales de la década de los años 90 por el empresario Kevin Ashton. Ashton fue uno de los fundadores del Auto-ID Center del MIT y formó parte del equipo que descubrió cómo vincular objetos a Internet a través de una etiqueta RFID. Fue el primero en usar la frase “Internet de las Cosas” en una presentación realizada en 1999.

El concepto de IoT ha ido evolucionando conforme la red inalámbrica de Internet ha sido más penetrante, así como los sensores integrados se han vuelto más precisos y la sociedad ha comenzado a entender la tecnología como una herramienta personal además de profesional.

La estandarización del IoT tendrá un impacto económico y social altísimo.

El IoT puede aportar numerosos beneficios como:

- Aumento de la eficiencia operativa: debido a que se ha empezado a automatizar los procesos de fabricación y a monitorear y controlar de manera remota e inalámbrica todas las operaciones.

- Se abre la puerta a nuevos modelos de negocio e ingresos. Se está alterando los modelos de negocios tradicionales y las empresas empiezan a crear servicios nuevos sobre la base de información y datos de sensores en tiempo real.
- Se pueden ofrecer productos de mejor calidad y de mayor rendimiento que permiten personalizar mejor al producto en función del cliente. Esto a su vez puede atraer a más clientes.

Según los datos recogidos en la página web de la empresa SAP SE (SAP, s.f.) (empresa multinacional alemana dedicada al diseño de productos informáticos de gestión empresarial):

- Unos 20.4 billones de dispositivos inteligentes estarán conectados a IoT para el 2020.
- El potencial impacto económico anual del IoT para 2025 será de unos $11T.
- El 50% de todos los activos operativos nuevos serán autosustentables para el 2020.

El internet de las cosas puede tener innumerables aplicaciones tecnológicas. La domótica sería una de ellas, mediante la distribución de una serie de sensores y procesadores se podría automatizar las luces del hogar, la temperatura, ventanas, etc. Además, se podrían controlar de forma inalámbrica desde el lugar que queramos.

Ilustración 2: Domótica

Otros elementos cotidianos como relojes, pulseras o camisetas se pueden hacer inteligentes gracias a su conexión con internet que intercambian datos con el servidor y actúan en función de la información que recogen.

Otro de los sectores que se puede ver beneficiado por el IoT sería el de la automoción, con los nuevos coches autónomos. En la industria también puede llegar a ser muy importante a la hora de la supervisión de máquinas, alumbrados inteligentes, huertos automatizados etc. En el sector del transporte se utilizan miles de sensores IoT como

en los aviones, trenes, buques y vehículos para optimizar todo, desde el rendimiento de motores y seguridad hasta logística y gestión de la cadena de suministro.

2.2. COMUNICACIONES INALÁMBRICAS

Como ya se ha mencionado anteriormente, el IoT se basa en la interconexión de los distintos dispositivos entre sí, y para llevarlo a cabo se necesita que la información se transmita por medio inalámbrico (sin cables), para también poder enviar esa información a internet y que pueda ser procesada por otro dispositivo u otro software de gestión.

Algunas de estas tecnologías inalámbricas que hemos podido estudiar para ver la que más nos podría interesar incorporar para nuestro proyecto serían:

2.2.1. Radio Frequency Identification (RFID).

Esta tecnología se introdujo inicialmente para la identificación y seguimiento de objetos a través de pequeños chips electrónicos denominados etiquetas. Por ejemplo, el pago automático en distintos sistemas como algunos peajes.

Ilustración 3: Logo RFID

Se componen de un circuito integrado (almacenar y procesar información), un transductor radio y una antena (Valencia, s.f.).

Existen tres tipos:

- **Pasiva**: No tienen fuente de alimentación (batería) y por lo tanto no pueden transmitir información. Se activan a través del lector RFID y transmiten solo una pequeña información como el identificador del objeto.
- **Activa**: Tiene su propia batería y pueden transmitir información de forma continua.
- **Híbrida (BAP)**: Lleva batería, pero solo transmite información en presencia de un lector RFID. La batería ayuda a transmitir la información en distancias más largas que la del tipo pasiva.

La información que este dispositivo envía es un identificador único para cada objeto.

Debido a su bajo coste, alta movilidad y la eficiencia en los dispositivos es una tecnología a tener en cuenta para el IoT.

Como inconvenientes para el IoT no podemos comunicar con Internet directamente, no puede establecer conexión a través de una puerta de enlace. Además, la necesidad de que el dispositivo de lectura esté próximo es otro inconveniente.

Podemos conectar un dispositivo RFID con Arduino a través del puerto serie.

2.2.2. Bluetooth y Bluetooth LE (Low Energy)

El Bluetooth es una tecnología estándar para el intercambio de datos a través de distancias cortas creando redes de área personal. Es una tecnología comúnmente utilizada en la actualidad ya que todos los smartphones la incluyen en los mismos y se suele utilizar para la conexión de auriculares, para la transmisión de información y archivos entre móviles, etc.

Ilustración 4: Logo Bluetooth

Se caracteriza por usar transmisiones de radio de onda corta en la banda de 2400 hasta 2480 MHz. Se suele utilizar para la conexión entre pares de dispositivos (Castillo, s.f.).

El Bluetooth LE se diferencia del Bluetooth clásico en que está diseñado para que el consumo de energía y el coste sean bastante reducidos, aunque mantiene un alcance de comunicación similar. El Bluetooth LE se suele aplicar en dispositivos que requieran un consumo bajo en energía, pequeños y de bajo costo, como relojes, sensores deportivos, teclados inalámbricos etc.

El Bluetooth es eficaz para el IoT debido a que ha sido uno de los primeros protocolos de comunicación inalámbricas con un consumo bajo de energía. Sin embargo, no se puede conectar directamente a Internet, necesitamos un intermediario, un PC por ejemplo, que actuará como conexión a Internet. Además, El bluetooth admite una subred con un máximo de 8 nodos.
Respecto a Arduino existen estos dos casos para la conexión por Bluetooth:

- Módulos de Bluetooth
  Existen números módulos bluetooth para incluir en un Arduino. (Barrera, 2018). Los módulos más frecuentes en el mercado son los módulos HC-06 y HC-05, que son muy económicos.
- Arduino 101
  (Arduino P. o., Genuino 101, s.f.) Este microcontrolador de Arduino llamado Genuino 101 lleva ya integrado el Bluetooth en la placa.

2.2.3. Enlaces de radiofrecuencia (RF Links)

Otra opción es conectar dispositivos mediante interfaces de radio frecuencia simple (Ruesca, s.f.). Tienen las siguientes características:

- Es ideal cuando las dimensiones son importantes, ya que es pequeño y barato.
- Proporciona un alcance de comunicaciones entre 100m y 1km dependiendo de la potencia de transmisión y de la antena que se utilice.
- Alcanza una velocidad de datos hasta 1 Mbps.
- Estos módulos se pueden conectar a microcontroladores como Arduino a través del puerto serie, pero no proporcionan ninguna aplicación del protocolo de comunicación TCP/IP. De nuevo tenemos problemas para conectarlos a Internet.

2.2.4. Redes móviles

Con estas redes se puede de acceder a Internet desde un dispositivo móvil, como un smartphone, tablet, portátil, … a través de una red de banda ancha. Algunas de sus características son:

- Hay disponibles muchos estándares con esta tecnología:
  - GPRS (80 Kbps)
  - 3G
  - 4G
- Proporcionan conectividad directa a Internet a diferentes velocidades según la tecnología. Desde 80 Kbps de la GPRS a unos pocos Mbps de la 3G y 4G.
- Debido a la complejidad del protocolo, la codificación de la información y los requisitos de señal alta de recepción de esta tecnología tiene un alto consumo de batería.
- Es una buena opción para el IoT ya que conectamos directamente con Internet.
- En Arduino hay disponibles varios módulos con GPRS.

2.2.5. ZigBee

Es un conjunto de protocolos de alto nivel de comunicación. Se utiliza para la radiodifusión digital de datos buscando ahorrar lo máximo posible de energía. Se basa en el estándar IEEE 802.15.4.

Ilustración 5: Logo Zigbee

Es un estándar abierto para abordar las necesidades únicas de bajo costo, redes inalámbricas de bajo consumo para la comunicación entre dispositivos (maquina a máquina o redes M2M) (Gutierrez, 2015).

Es muy similar al Bluetooth aunque se diferencia en los siguientes aspectos:

- Admite más nodos (más dispositivos en una red). Una red Zigbee puede constar de un máximo de 65535 nodos distribuidos en subredes de 255 nodos.
- Mejor eficiencia energética (Menor consumo).
- Mayor alcance (300m).
- Menor velocidad que el Bluetooth. Bluetooth: 3000 Kbps y ZigBee: 250 Kbps. Por eso este sistema no se ha llevado a los móviles y se utiliza en la comunicación entre dispositivos (por ej. Domótica).

Al igual que Bluetooth requiere un dispositivo con acceso a Internet (un PC).

Respecto a su uso con Arduino, están disponibles los Módulos XBee.

2.2.6. WiFi

En este tipo de red inalámbrica se analizará con más detalle debido a que será la escogida para la comunicación entre los dispositivos de este trabajo. Más adelante se explicará la razón de esta elección para el proyecto.

WiFi es una tecnología que permite la interconexión de manera inalámbrica de distintos dispositivos electrónicos. Estos dispositivos pueden ser de todo tipo (teléfonos, ordenadores, videoconsolas, smartwatches, reproductores de música, etc.) Estos dispositivos pueden conectarse entre sí o a internet mediante un punto de acceso de red inalámbrica.

Ilustración 6: Logo WiFi

Wi-Fi es una marca de la Alianza Wi-Fi, la organización comercial que adopta, prueba y certifica que los equipos cumplen con los estándares 802.11 relacionados con redes inalámbricas de área local.

Existen varios dispositivos WiFi, los cuales se pueden dividir en dos grupos: dispositivos de distribución o de red, entre los que destacan los enrutadores, puntos de acceso y repetidores; y dispositivos terminales que en general son las tarjetas receptoras para conectar a la computadora personal, ya sean internas (tarjetas PCI) o bien USB.

2.2.6.1. Tipos de WiFi

Hay diversos tipos de WiFi, todos ellos basados en el estándar IEEE 802.11.
Estos tipos de WiFi están diferenciados y explicados en el artículo "Conoce las diferencias entre los tipos de WiFi" de Manuel Ramírez (Ramírez, 2018):

- **IEEE 802.11**

  Es el primero de todos y fue creado en 1997. En su momento fue llamativo por una conexión máxima de 11 megabits por segundo. Al ser primero también significa que actualmente no hay dispositivos que puedan utilizarlo, pero es interesante tenerlo en cuenta para repasar un poco la historia de los distintos estándares WiFi.

- **IEEE 802.11a**

  Creado en 1999, este tipo de WiFi funciona en banda 5 GHz. Su objetivo fue intentar encontrar las mínimas interferencias posibles, ya que eran muchos los dispositivos que utilizaban la banda 2.4 GHz.

  Estamos ante un tipo de WiFi que es muy rápido al ofrecer ratios de datos que llegan a los 54 megabits por segundo. Hay que citar que la frecuencia de 5GHz tiene más dificultades con objetos que se encuentren en su trayectoria de señal, por lo que el rango es bastante pobre.

  Lo podremos encontrar generalmente en redes de negocios, mientras la 802.11b, la que a continuación detallamos, es más frecuente en el hogar.

- **IEEE 802.11b**

Fue creado también en 1999 y es un tipo de WiFi que usa la banda más típica 2.4 GHz. Puede llegar a obtenerse una velocidad máxima de 11 megabits por segundo. Estamos ante el estándar que fue capaz de poner la tecnología WiFi en su máximo apogeo.

Tiene un problema de interferencia con teléfonos inalámbricos, microondas y otros dispositivos que utilicen la banda 2.4 GHz.

- **IEEE 802.11g**

  Llegó más tarde, en 2003. El tipo de WiFi 802.11g pasa a ofrecer un ratio de datos de hasta 54 megabits por segundo a través del uso de banda 2.4 GHz. Estamos ante el estándar que fue capaz de propagar el uso de WiFi.

  Incluso a día de hoy es un tipo de WiFi bien común, ya que es rápido y los routers que se valen de él son increíblemente baratos. Al ser compatible con el estándar 802.11b, los puntos de acceso creados valen para redes 802.11b.

- **IEEE 802.11n**

  Es de los tipos de WiFi más rápidos y actuales, y que se introdujo en 2009. Aunque su penetración en el mercado ha sido lenta, ahora se puede encontrar en todo tipo de routers y portátiles.

  Se caracteriza por operar tanto a 2.4GHz y 5GHZ, aparte de ofrecer soporte a multi-canal. Esos canales permiten alcanzar ratios de datos de 150 megabits por segundo, lo que significa que podemos acceder a velocidades de 600 megabits por segundo.

- **IEEE 802.11AC**

  La nueva señal WiFi que llega a alcanzar velocidades de hasta 1.300 megabits por segundo en 5 GHZ y hasta 450 Mbps en 2.4 GHz. Se vale de la tecnología dual band para conectarse con banda 2.4 GHz y 5 GHz.

  Los teléfonos más modernos ya tienen este estándar y pueden alcanzar estas velocidades si el router también lo posee.

2.2.6.2. Seguridad y fiabilidad

Existen varias alternativas para garantizar la seguridad de estas redes. Las más comunes son la utilización de protocolos de cifrado de datos para los estándares WiFi que se encargan de codificar información transmitida para proteger su confidencialidad, proporcionados por los propios dispositivos inalámbricos. La mayoría de las formas son las siguientes:

- WEP, cifra los datos en su red de forma que solo el destinatario deseado pueda acceder a ellos. Los cifrados de 64 y 128 bits son dos niveles de seguridad WEP. WEP codifica los datos mediante una “clave” de cifrado antes de enviarlo al aire. Este tipo de cifrado no está recomendado debido a las grandes vulnerabilidades que presenta

ya que cualquier *cracker* puede conseguir sacar la clave, incluso aunque esté bien configurado y la clave utilizada sea compleja.

- WPA: presenta mejoras como generación dinámica de la clave de acceso. Las claves se insertan como dígitos alfanuméricos.
- IPSEC (túneles IP) en el caso de las VPN y el conjunto de estándares IEEE 802.1X, que permite la autenticación y autorización de usuarios.
- Filtrado de MAC, de manera que solo se permite acceso a la red a aquellos dispositivos autorizados. Es lo más recomendable si solo se va a usar con los mismos equipos y si son pocos.
- Ocultación del punto de acceso: se puede ocultar el punto de acceso (*router*) de manera que sea invisible a otros usuarios.
- El protocolo de seguridad llamado *WPA2* (estándar 802.11i), que es una mejora relativa a WPA. En principio es el protocolo de seguridad más seguro para Wi-Fi en este momento. Sin embargo, requieren *hardware* y *software* compatibles, ya que los antiguos no lo son.

2.2.6.3. Ventajas y desventajas del WiFi

Ventajas

- Se pueden conectar a esta red una gran cantidad de dispositivos en un espacio bastante amplio.
- La Alianza Wi-Fi asegura que la compatibilidad entre dispositivos con la marca Wi-Fi es total, con lo que en cualquier parte del mundo podremos utilizar la tecnología WiFi con una compatibilidad absoluta.
- La principal ventaja sobre el resto de redes inalámbricas es que se puede conectar directamente a internet, además de que esta conexión es muy fácil de hacer.

Desventajas

- Menor velocidad que una conexión cableada debido a las inferencias y pérdidas de señal en el ambiente.
- Problemas en la seguridad a la hora de que pueden acceder a la contraseña. Por ello es recomendable utilizar el estándar WPA2.
- No es compatible con otros tipos de conexiones inalámbricas.
- La potencia de la señal WiFi se ve afectada por agentes físicos del ambiente.
- Respecto a las otras tecnologías inalámbricas consume más energía.

2.3. APLICACIONES WEB

Una aplicación web o *web app* es la desarrollada con lenguajes muy conocidos por los programadores, como es el HTML, Javascript y CSS. La principal ventaja con respecto a la nativa es la posibilidad de programar independiente del sistema operativo en el que se usará la aplicación. De esta forma se pueden ejecutar en diferentes dispositivos sin tener que crear varias aplicaciones.

Las aplicaciones web se ejecutan dentro del propio navegador web del dispositivo a través de una URL. Por ejemplo en Safari, si se trata de la plataforma iOS. El contenido se adapta a la pantalla adquiriendo un aspecto de navegación APP.

En realidad, la gran diferencia con una aplicación nativa es que no necesita instalación por lo que no pueden estar visibles en app store y la promoción y comercialización debe realizarse de forma independiente. De todos modos, se puede crear un acceso directo que sería como "instalar" la aplicación en el dispositivo.
Las apps web móviles son siempre una buena opción si nuestro objetivo es adaptar la web a formato móvil.

Ventajas e inconvenientes de las aplicaciones web

VENTAJAS	INCONVENIENTES
- **Utiliza lenguajes muy conocidos como HTML y Css.**	- Necesidad de conexión a internet obligatoriamente.
- **Se ejecutan dentro del propio navegador web a través de una URL**	- Pierda visibilidad al no poder encontrarla en los distintos stores.
- **No necesita ser instalada desde las tiendas App, como Google Play Store o Apple App Store**	- La web deberá estar programada para su reproducción en cualquier sistema operativo.
- **Se pueden reproducir en cualquier sistema operativo**	- Se tendrá algunas restricciones de acceso a algunas características del hardware de tu dispositivo
- **No necesita una actualización. En este sentido, la modernización la debe sufrir la propia web a la que está enlazada la aplicación.**	- La inversión que uno se puede ahorrar en el desarrollo de la aplicación se tiene que hacer en mejorar la página web. No sirve de nada hacer una Web App si no adaptas tu página para que se pueda reproducir con calidad en cualquier dispositivo.
- **Su beneficio más importante, el precio. El desarrollo de una Web App es el más económico. Consiste en crear un enlace o varios desde aplicación a una URL.**	

2.4. JUSTIFICACIÓN EN LA APLICACIÓN PARA EL PROYECTO

2.4.1 Comunicación Inalámbrica

Analizando las posibles opciones de comunicación inalámbrica entre nuestros dispositivos en el proyecto, se ha decidido en emplear placas Arduinos con conexión WiFi por las siguientes razones:

1. La conexión WiFi es bastante fácil de establecer.
2. En Arduino existen módulos y placas ya que disponen de WiFi a un precio bastante bajo.
3. Hoy en día hay redes WiFi en prácticamente todos los espacios de las ciudades, tanto en zonas públicas como privadas, por lo que se podría instalar nuestro sistema en prácticamente cualquier lado.
4. Las redes WiFi nos proporcionan acceso directo a internet, cosa que otras tecnologías no nos ofrecen, además de que el alcance es muy amplio si los módulos están conectados a la misma red WiFi.

La única desventaja sería que estos dispositivos consumen más energía que el resto de tecnologías inalámbricas.

2.4.2 Aplicación web

Teniendo en cuenta las características y analizando las ventajas y desventajas de las aplicaciones web se ha elegido este método debido a que básicamente se puede reproducir desde cualquier plataforma (solo se necesita un navegador web) y además también se puede ejecutar desde cualquier localización mediante la IP.

Como nuestro proyecto está dirigido a un centro de mayores (no necesita promoción), no necesita estar en ninguna plataforma como Google Store, por lo que la aplicación web es válida en este caso también. Además, no es necesario una actualización de la misma salvo que se quiera modificar algún detalle del sistema, algo que es muy poco probable.

Por último, económicamente es muy accesible ya que sólo hay que crear un enlace desde la aplicación a una URL.

3. MATERIAL Y MÉTODOS

3.1 ARDUINO ETHERNET

3.1.1. ESPECIFICACIONES TÉCNICAS

El Arduino Ethernet es una placa de microcontrolador basada en el ATmega328. Se utilizará este microcontrolador como servidor en este proyecto.

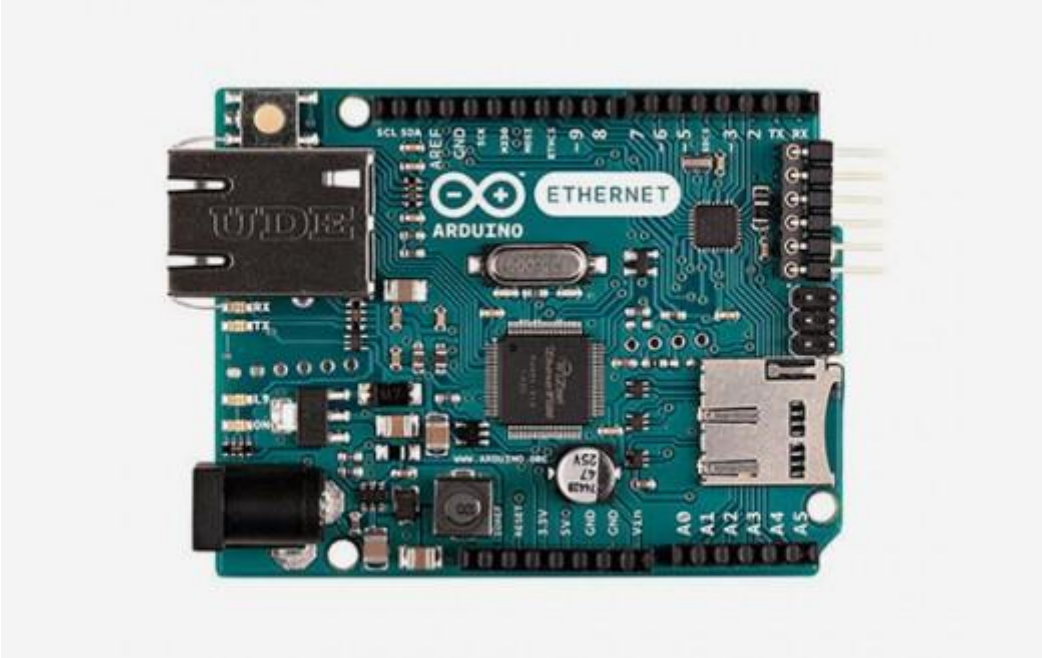


Ilustración 7: Arduino Ethernet

Está basada en el chip ethernet Wiznet W5100. El Wiznet W5100 provee de una pila de red IP capaz de soportar TCP y UD.

Los protocolos UD y TCP son usados para transmitir datos, o paquetes de información, a través de redes basadas en IP. TCP está orientado a conexiones, donde una vez establecida la misma, la información se puede transmitir en ambas direcciones. UD es un protocolo de Internet más sencillo, sin necesidad de conexiones. Con él se pueden enviar múltiples mensajes en grupos (paquetes) de datos. UD es útil para aplicaciones que necesitan transmisión rápida y efectiva. La capacidad de transferencia sin conexiones de UD le hace útil para servidores que reciben una gran cantidad de peticiones pequeñas de un alto número de clientes.

Usa la librería Ethernet para leer y escribir los flujos de datos que pasan por el puerto ethernet.

Las especificaciones técnicas del Arduino Ethernet son las siguientes:

Microcontrolador	ATmega328
Tensión de funcionamiento	5V
Enchufe de voltaje de entrada (recomendado)	7-12V
Enchufe de voltaje de entrada (límites)	6-20V
Voltaje de entrada PoE (límites)	36-57V
Pines digitales de E / S	14 (de los cuales 4 proporcionan salida PWM)
Arduino Pins reservados:	
	10 a 13 utilizados para SPI
	4 utilizados para la tarjeta SD
	2 interrupciones W5100 (cuando puenteado)
Entradas analógicas	6
Corriente DC por Pin de E / S	40 mA
Corriente DC para 3.3V Pin	50 mA
Memoria flash	32 KB de los cuales 0.5 KB utilizados por el cargador de arranque
SRAM	2 KB
EEPROM	1 KB
Velocidad de reloj	16 MHz
Controlador de Ethernet integrado W5100 TCP / IP	
Tarjeta micro SD, con traductores de voltaje activo.	
Longitud	68.6 mm
Anchura	53.3 mm
Peso	28 gr

El Pinout del Arduino Ethernet es el siguiente:

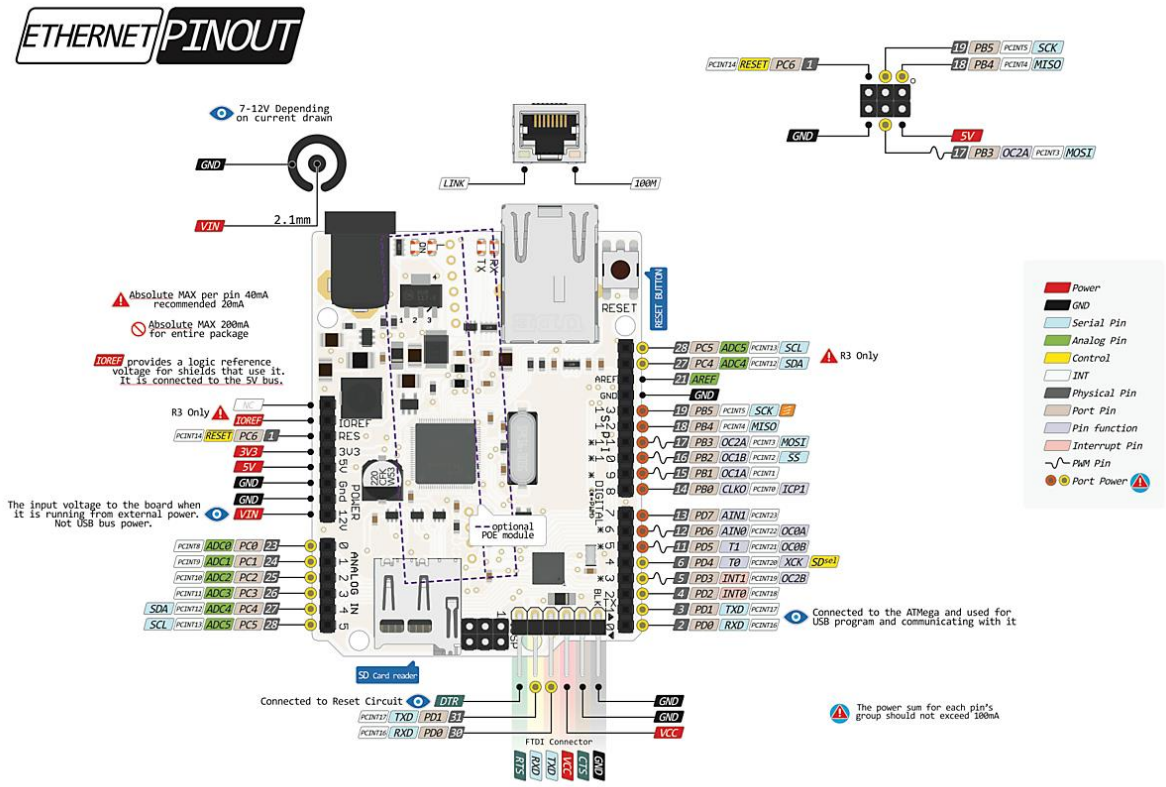


Ilustración 8: Pinout Arduino Ethernet

A modo de resumen los aspectos más importantes de las especificaciones de este Arduino que vamos a tener más en cuenta van a ser:

- Su alimentación es de 5V.
- Tiene 14 pines de entradas/salidas digitales (4 proveen salida de PWM)
- 6 salidas análogas
- Posee un oscilador de 16 Mhz de cristal.
- Un conector Jack con una conexión RJ45.
- Un botón de reset.
- La corriente DC por cada pin de entrada/salida es de 40mA.
- Tarjeta Micro SD (Pin 4).

Además, se incluyen algunos LEDs para darnos la siguiente información:

- ON: indica que la placa está alimentada.

- LINK: indica la presencia de una conexión de red y parpadea cuando el microcontrolador envía o recibe datos.
- 100M: indica la presencia de una conexión de red de 100 Mb/s (de forma opuesta a una de 10Mb/s).
- RX: parpadea cuando se recibe datos.
- TX: parpadea cuando se envía datos.

3.1.2 ALIMENTACIÓN

La alimentación externa del microcontrolador puede llegar a través de una fuente de alimentación con adaptador AC a DC con un suministro externo de 6 a 20 voltios a través de un conector de 2.1mm de centro positivo al conector de energía Jack del Arduino. Si se suministra menos de 7V, el pin de 5V puede que no llegue a suministrar los 5V y puede hacer que el microcontrolador funcione de manera inestable, por lo que se recomienda una alimentación de entre 7 y 12 voltios.

Los pines de energía son los siguientes:

- VIN. La entrada de voltaje de la tarjeta Arduino cuando se está usando una fuente de energía externa (opuesto a los 5V desde la conexión USB u otra fuente de energía regulada. Puedes darle voltaje a través de este pin o, si suministras voltaje vía el conector Jack, puedes acceder a él por este pin).
- 5V. Este pin proporciona una salida regulada de 5V. La tarjeta puede tomar energía desde el conector Jack de energía DC (7-12V), el conector USB (5V) o el pin WIN de la tarjeta (7-12V).
- 3.3V. Suministro de 3.3 volts generado por el regulador incorporado. La corriente máxima generada es de 50mA.
- GND. Pines de tierra.
- También es posible añadir un módulo PoE, que está diseñado para extraer energía desde un cable Ethernet categoría 5 de par trenzado:
    - Certificado por la IEEE802.3af.
    - Baja salida de ruido y onda (100mVpp).
    - Rango de entrada de voltaje de 36V a 57V.
    - Protección de sobrecarga y cortocircuito.
    - Salida de 9V.
    - Convertidor DC/DC de alta eficiencia: tipo 75% @ 50% de carga.
    - Aislamiento de 1500V (entrada a salida).

En este caso en particular se ha empleado un módulo adaptador convertidor (5V o 3.3V) llamado FT232RL FTDI. Este módulo es un convertidor de RS232 a USB, a través de él se puede alimentar a 5V la placa y además debido a su conector miniUSB se puede conectar con el ordenador para subir el programa a la placa.

Además este módulo tiene unos pines que se puede configurar en función de nuestras necesidades (3.3V, el reloj, RX, Tx…). Este módulo irá conectado a los pines FTDI del Arduino Ethernet.

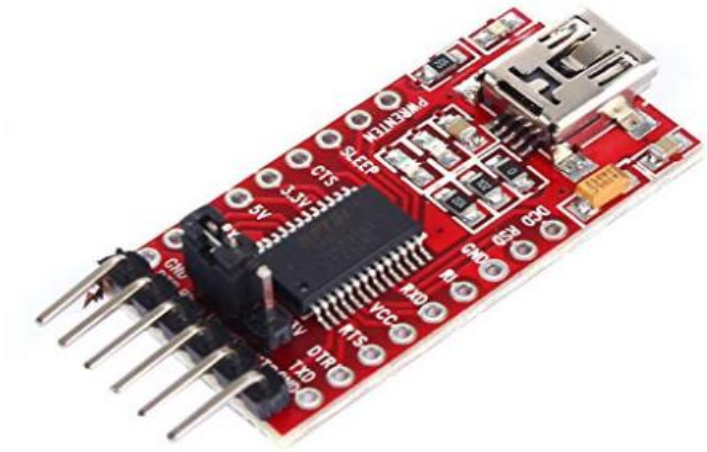


Ilustración 9: FT232RL

3.1.3 MEMORIA

El ATmega328 tiene 2KB de SRAM y 1KB de EEPROM y una memoria Flash de 32KB, de los cuales 0.5KB son usados por el gestor de arranque.

- La memoria Flash es una memoria de contenido permanente en la que se carga de forma permanente el programa que se quiere ejecutar hasta que lo sustituyamos por otro.
- La memoria SRAM es una memoria volátil. En esta memoria se almacenarán los datos y variables que utilizará el programa en el tiempo de ejecución de este. El contenido de la memoria se elimina cuando se deje de alimentar la placa o cuando la reiniciemos.
- La memoria EEPROM es una memoria de contenido permanente donde se puede almacenar los datos que queremos conservar aunque se deje de alimentar la placa, por lo que se podrían utilizar en otras ejecuciones.

Se pueden añadir memoria SRAM y EEPROM conectando módulos adicionales y comunicándolos con el micro mediante algún protocolo de comunicación como por ejemplo I2C. Además, se puede incluir una tarjeta SD para aumentar la memoria Flash.

3.1.4 COMUNICACIÓN

El Arduino Ethernet tiene una gran facilidad para comunicarse con cualquier medio usando el hardware adecuado. Un listado de las comunicaciones para Arduino sería el siguiente:

- Serial/UART
- I2C
- SPI
- one wire
- RS232
- RS485
- CAN BUS y cable
- NFC
- ANT+
- RF (433MHz, etc…)
- Bluetooth
- IR
- Power Line (modulo linksys SPI)
- Ethernet
- WiFi (b/g/n/ac/…)
- GSM (GPRS/HSPA/LTE/NB-IoT/LTE-IoT…) XBee/ZigBee
- nRF24

Para realizar cualquiera de estas comunicaciones necesitaremos incluir las librerías correspondientes, en este caso se incluirá la librería Ethernet.h y en el Arduino Ethernet y la librería WiFi.h para el Arduino MKR1000. De las librerías se hablará en apartados posteriores.

3.2 ARDUINO WIFI MKR1000

El Arduino MKR1000 es un microcontrolador de Arduino que fue lanzado a finales del año 2015 y destaca porque lleva integrado WiFi además de su pequeño tamaño, la hacen perfecta para realizar proyectos de IoT.

Ilustración 10: ArduinoWiFi MKR1000

3.2.1 ESPECIFICACIONES TÉCNICAS

Este Arduino está basado en la MCU ATSAMW25 .

Este chip está compuesto por tres bloques principales.

- El microcontrolador ARM SAMD21 Cortex-M0+ de 32 bits y de bajo consumo.
- El módulo WiFi WINC1500 de 2,4 GHz y de bajo consumo.
- El módulo de encriptación de datos por hardware ECC508, para que nuestras comunicaciones sean seguras.

Las características técnicas más importantes de este Arduino se recogen en la siguiente tabla:

Microcontrolador:	ARM SAMD21 Cortex-M0+ de 32 bits
Junta de fuente de alimentación:	(USB / VIN) 5V
Batería soportada sola célula	3.7V , mínimo (*) Li – Po 700 mAh
Voltaje de circuito operativo	3.3V
Digital pines I / O:	8
PWM Pins	4 (D2 – D5)
UART:	1
SPI:	1
I2C:	1
Entrada analógica pines:	7 (ADC 8/10/12 bits)
Salida analógica patillas	1 (DAC de 10 bits)
Interrupciones externas:	8
Corriente continua para Pin I/O:	7 mA
EEPROM	No
Memoria flash:	256 KB
SRAM:	32KB
Velocidad de reloj	32,768 kHz , 8 MHz y 48 MHz
Full-Speed USB Device y Host incrustado	

El pinout del Arduino WiFi MKR1000 es el siguiente:

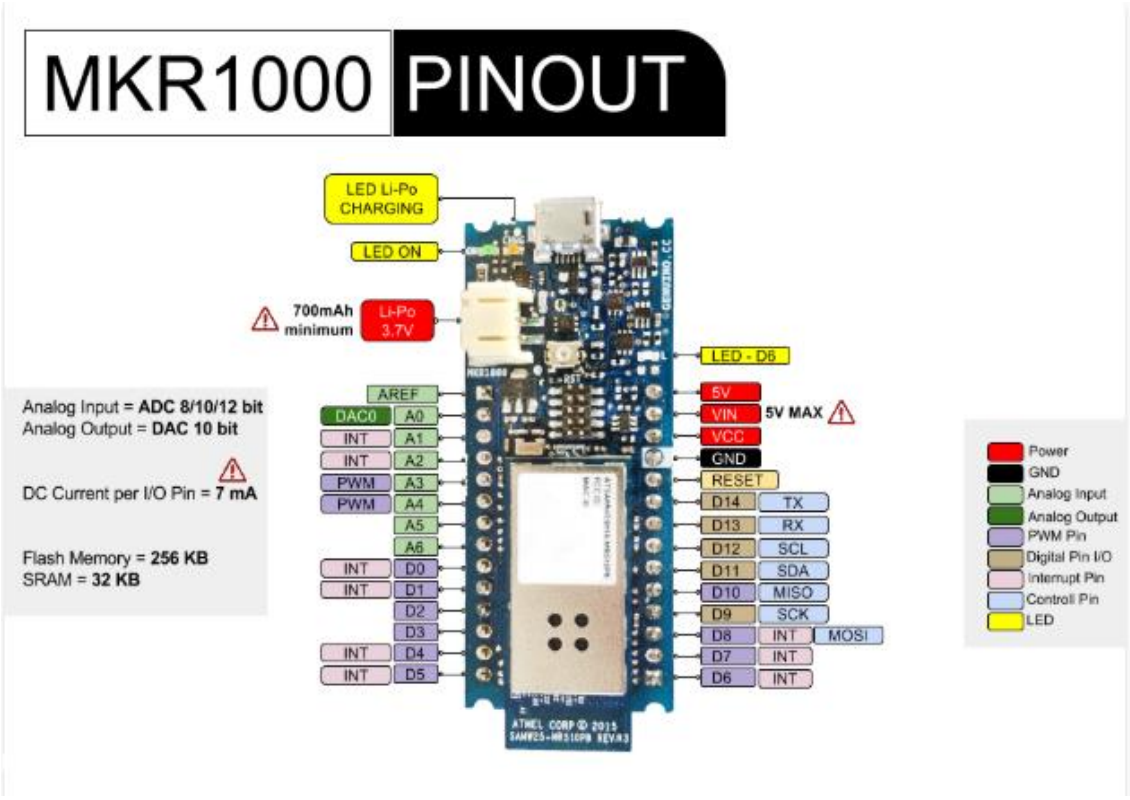


Ilustración 11: Pinout Arduino WiFi MKR1000

Al igual que el Arduino Ethernet esta placa también posee algunos LEDs que nos indican si está alimentada (LED ON), si trasmite datos (LED TRX) o si recibe datos (LED RX).

3.2.2 ALIMENTACIÓN

Al Arduino MKR1000 se le puede suministrar energía de tres formas diferentes:

- A través del puerto micro USB, conectándolo al PC o bien a una fuente que proporcione 5V.
- A través del pin VIN. También se requiere una diferencia de potencial de 5V.
- Se puede hacer uso de baterías ya que dispone de pines para la conexión de las mismas. Se recomienda que se utilice una batería de 3.7V y de mínimo 700 mAh.

Hay que tener precaución con el voltaje que se le suministren a las entradas ya que funcionan con 3.7V, por lo que suministrar el valor típico de 5V podría dañar la placa.

Además, el consumo de este Arduino se puede dividir en dos. Uno sería el del microcontrolador que es de alrededor de 20mA y por otro lado el del módulo WiFi, que es de unos 100mA.

3.2.3 MEMORIA

Dispone de una memoria Flash de 256 Kb y una memoria SRAM de 32 Kb. Aunque no dispone de memoria EEPROM no supone mucho problema debido a su conectividad. Estas memorias ya se explicaron en la parte del Arduino Ethernet.

3.2.4 COMUNICACIÓN

Como ya hemos nombrado anteriormente en el Arduino Ethernet, esta placa posee numerosas formas de comunicación, pero se tratará solo la parte de la comunicación vía WiFi y en la librería WiFi.h que se hablará más adelante.

3.3 DISPLAY 7 SEGMENTOS

Un display es un componente electrónico comúnmente utilizado para representar números y letras en muchos proyectos electrónicos. Se componen de 7 Ledes que hacen la forma de un "8", y estos dispositivos luminosos se pueden controlar para representar el número o letra que deseemos.

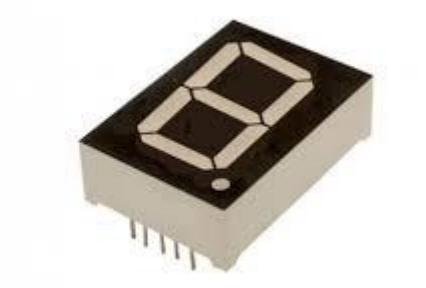


Ilustración 12: Display 7 segmentos

Hay dos tipos de displays de 7 segmentos: los de ánodo común y el de cátodo común.

Lo que les diferencia a uno del otro es que en el del cátodo común, el punto circuital en común para todos los leds es el cátodo (gnd, 0V), mientras que para el del ánodo común el punto de referencia es Vcc (5V). Esto es un dato esencial a la hora de poner en funcionamiento este componente debido a que los displays de cátodo común se controlarán con voltaje positivo (1 lógicos) y los de ánodo común se controlarán con 0 lógicos.

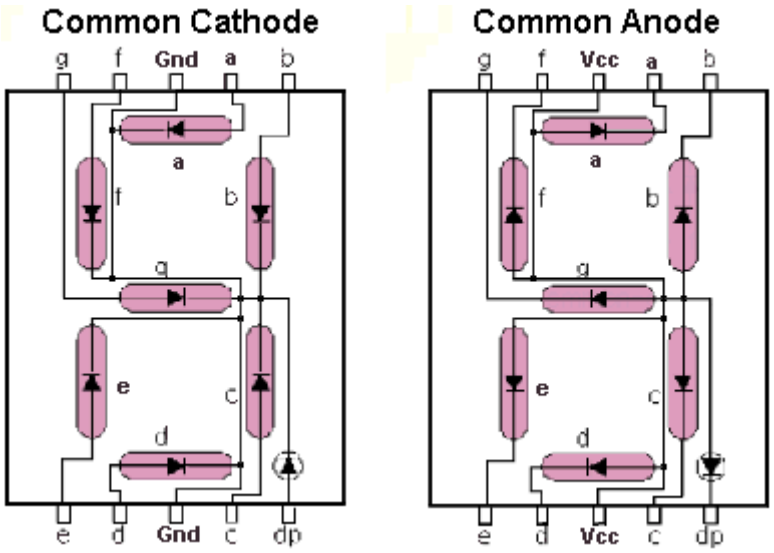


Ilustración 13: Tipos de display 7 segmentos

Estos Ledes trabajan con corrientes y tensiones muy bajas por lo que se podrían conectar directamente a los pines del microcontrolador, pero es recomendable que se conecte una resistencia en serie entre el pin del display y la salida del microcontrolador para regular la corriente y aumentar la vida útil del dispositivo. La intensidad lumínica dependerá del valor de la resistencia empleada.

En este proyecto he empleado el display D5611 , que es de ánodo común.

A la hora de controlar el número que se quiere representar en el display es sencillo. Solo tenemos que hacer uso de la imagen anterior donde se le asigna una letra a cada led, por lo que al poner cada pin a nivel alto o bajo, dependiendo si es de ánodo o cátodo común, se iluminará un segmento del display. En la siguiente tabla de verdad se puede observar cada caso correspondiente.

	Catodo Comun							
	Numero	A	B	C	D	E	F	G
Enable	0	1	1	1	1	1	1	0
0	1	0	1	1	0	0	0	0
0	2	1	1	0	1	1	0	1
0	3	1	1	1	1	0	0	1
0	4	0	1	1	0	0	1	1
0	5	1	0	1	1	0	1	1
0	6	1	0	1	1	1	1	1
0	7	1	1	1	0	0	0	0
0	8	1	1	1	1	1	1	1
0	9	1	1	1	1	0	1	1

	Anodo Comun							
	Numero	A	B	C	D	E	F	G
Enable	0	0	0	0	0	0	0	1
1	1	1	0	0	1	1	1	1
1	2	0	0	1	0	0	1	0
1	3	0	0	0	0	1	1	0
1	4	1	0	0	1	1	0	0
1	5	0	1	0	0	1	0	0
1	6	0	1	0	0	0	0	0
1	7	0	0	0	1	1	1	1
1	8	0	0	0	0	0	0	0
1	9	0	0	0	0	1	0	0

Ilustración 14: Tablas de verdad Displays

En este caso se hará uso de la tabla de la derecha, la del ánodo común. El pin de Enable a 1 es el de Vcc del display. Como se puede observar por ejemplo en el caso del número 0 se pondrían a nivel bajo todos los segmentos menos la G debido a que se activan a nivel bajo.

3.4 LEDS

Un Led es un diodo que emite luz al pasar la corriente a través de él. Sólo permite el paso de corriente en el sentido ánodo-cátodo. Para ello debe quedar correctamente polarizado, esto es, la tensión ánodo-cátodo debe ser superior a un umbral especificado por el fabricante.

Ilustración 15: Diodos Led

3.4.1 CARACTERÍSTICAS

Los Diodos Leds tienen dos patillas de conexión: ánodo y cátodo. Si no se ha manipulado el diodo, la patilla larga es el ánodo y la corta el cátodo. Para que se ilumine debe ser polarizado en directo (diferencia de potencial ánodo-cátodo superior al valor umbral).

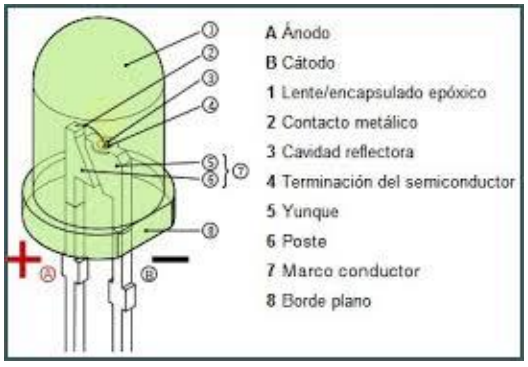


Ilustración 16: Características Diodos Led

Dependiendo del material del semiconductor, éste emitirá una luz de un color diferente.

3.4.2 VENTAJAS

Consumen menos energía que las bombillas convencionales debido a que los leds pierden mucha menos energía en forma de calor que las lámparas convencionales. Por lo que casi toda la energía se emplea en convertirla en luz y consume menos.

Además, otra ventaja es que la vida útil del led es mucho mayor que el de una bombilla normal. La del Led puede ser superior a cien mil horas de luz mientras que la de la bombilla dura unas cinco mil horas.

3.5 PULSADOR

Es un dispositivo electrónico que permite o no el paso de la corriente a través de él dependiendo si está pulsado o no.

Ilustración 17: Pulsador

El pulsador no es lo mismo que un interruptor, en este caso hay que mantener el botón presionado para que deje pasar la corriente a través de él. En el caso que no esté pulsado el circuito quedará abierto.

3.6 COMPONENTES SOFTWARE

- Software ARDUINO (lenguaje de programación en C, C++)

-Programación en HTML.

3.6.1 Software ARDUINO

3.6.1.1 Introducción

Según la página Web oficial Arduino:

> Arduino es una plataforma electrónica de código abierto basada en hardware y software fáciles de usar. Las placas Arduino pueden leer entradas, tanto analógicas como digitales y convertirlas en salidas. Además, puede decirle a su tarjeta qué debe hacer enviando una serie de instrucciones al microcontrolador de la tarjeta. Para realizar esto utiliza el lenguaje de programación Arduino (que se basa en Wiring, C++) y el software Arduino (IDE) basado en el procesamiento. (Arduino, s.f.)

Ilustración 18: Logo Arduino

Arduino nació en el Instituto de Diseño de Interacción Ivrea como una herramienta fácil para la creación rápida de prototipos, dirigida a estudiantes sin experiencia en electrónica y programación. Tan pronto como llegó a una comunidad más amplia, la placa Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y desafíos, diferenciando su oferta de tableros simples de 8 bits a productos para aplicaciones IoT, impresión portátil, impresión 3D y entornos integrados. Todas las placas Arduino son completamente de código abierto, lo que permite a los usuarios construirlas de forma independiente y, eventualmente, adaptarlas a sus necesidades particulares. El software también es de código abierto y está creciendo a través de las contribuciones de los usuarios de todo el mundo.

Las ventajas que ofrece Arduino se pueden resumir como las siguientes:

- Multiplataforma: El software de Arduino se puede ejecutar en Windows, Macintosh y Linux.

- Entorno de programación simple y claro.
- Software de código y extensible: Arduino se publica como herramientas de código abierto, disponible para la extensión por programadores experimentados. El lenguaje se puede expandir a través de las bibliotecas de C ++, y las personas que deseen comprender los detalles técnicos pueden dar el salto de Arduino al lenguaje de programación AVR C en el que se basa.
- Barato. La mayoría de los microcontroladores de Arduino cuestan menos de 50€.
- Fuente abierta y extensible. Se pueden crear nuevas versiones de los módulos, hasta ampliarlos y mejorarlos.

3.6.1.2 Lenguaje de Programación Arduino

El lenguaje de programación de Arduino, como se ha indicado anteriormente es una versión simplificada de C/C++, los programas de Arduino de denominan Sketch tiene extensión ".ino" y una vez compilados se convierten a código binario AVR.

Características:

Es un lenguaje case-sensitive (diferencia entre mayúsculas y minúsculas)
Todas las instrucciones terminan con un punto y coma. Un programa o sketch de Arduino se compone de tres secciones:

- Sección de declaraciones:

  En esta sección se declararán las variables globales del programa y también se incluirán las librerías y ficheros que utilicemos en el mismo.

- Sección "void setup() { … }" :

  Esta sección solo se ejecutará una vez cuando se alimente la placa o cuando se reinicie la misma. Se ejecutarán una sola vez todas las instrucciones dentro de la función setup(). Estas instrucciones incluirán las inicializaciones de las variables, así como también se establecerán el estado de los pines de la placa. Además, también en esta sección se establecerán las conexiones inalámbricas de cada Arduino.

- Sección "void loop() { … }" :

  Justo tras terminar la sección setup() se ejecutarán las instrucciones contenidas dentro de la función loop() de forma continua e infinitas veces hasta que la placa sea desconectada o reseteada.

3.6.2 LENGUAJE HTML

El lenguaje HTML es en realidad un lenguaje de marcado para la elaboración de páginas web.

Como en nuestro proyecto vamos a realizar un servidor web donde vamos a mostrar unos datos, debemos de explicar un poco en qué consiste este lenguaje de marcas de hipertexto.

Ilustración 19: introducción HTML

Según el artículo de la página web "definición.de" podemos introducir el HTML como se indica a continuación.

El lenguaje HTML se trata de un formato abierto que surgió a partir de las etiquetas SGML (Standard Generalized Markup Language). Concepto traducido generalmente como "Estándar de Lenguaje de Marcado Generalizado" y que se entiende como un sistema que permite ordenar y etiquetar diversos documentos dentro de una lista. Este lenguaje es el que se utiliza para especificar los nombres de las etiquetas que se utilizarán al ordenar, no existen reglas para dicha organización, por eso se dice que es un sistema de formato abierto.

EL HTML se encarga de desarrollar una descripción sobre los contenidos que aparecen como textos y sobre su estructura, complementando dicho texto con diversos objetos (como fotografías, animaciones, etc).

Es un lenguaje muy simple y general que sirve para definir otros lenguajes que tienen que ver con el formato de los documentos. El texto en él se crea a partir de etiquetas, también llamadas tags, que permiten interconectar diversos conceptos y formatos.

Para la escritura de este lenguaje, se crean etiquetas que aparecen especificadas a través de corchetes o paréntesis angulares: **<** y **>**. Entre sus componentes, los elementos dan forma a la estructura esencial del lenguaje, ya que tienen dos propiedades (el contenido en sí mismo y sus atributos).

Por otra parte, cabe destacar que el HTML permite ciertos códigos que se conocen como scripts, los cuales brindan instrucciones específicas a los navegadores que se encargan de procesar el lenguaje. Entre los scripts que pueden agregarse, los más conocidos y utilizados son JavaScript y PHP.

El marcado estructural es el que estipula la finalidad del texto, aunque no define cómo se verá el elemento. El marcado visual, por su parte, es el que se encarga de señalar cómo se verá el texto más allá de su función.

Para conocer el código HTML que utiliza una página web, hay que seleccionar Ver código fuente en nuestro navegador (como Internet Explorer o Mozilla Firefox). Al elegir esta opción, se abrirá el editor de texto con el código HTML de la página que se está visualizando.

(Pérez Porto y Gardey, publicado en 2008.Actualizado:2012)

3.7 JUSTIFICACIÓN DE NUESTRA ELECCIÓN

Hemos decidido realizar este prototipo con los microcontroladores de Arduino y su lenguaje de programación debido a algunas de las ventajas que nos proporciona como las siguientes:

- Es un lenguaje sencillo que se basa en la programación en C y C++, ya estudiado en el grado cursado por lo que ya se tenía bastante conocimiento del mismo.
- Arduino proporciona muchas librerías que facilita enormemente algunos aspectos de la programación.
- Con un microcontrolador de Arduino se puede realizar de forma bastante sencilla todos los montajes con los componentes que necesitamos para llevar a cabo este proyecto.
- El coste económico de los Arduino es asequible.

Respecto al código HTML, como se necesita programar una página web sencilla y el lenguaje HTML es fácil de aprender y de interpretar se empleará en nuestro servidor web.

Además, el lenguaje HTML es el más extendido y todos los navegadores lo admiten.

4. DESARROLLO DEL PROYECTO Y RESULTADOS

4.1 Descripción del proyecto

El proyecto se ubicaría en una residencia por lo que habría que distribuir los microcontroladores que se van a usar en las localizaciones específicas del edificio. Debido a esta circunstancia y la posible movilidad futura de los sensores, para la realización del piloto se opta por la utilización de placas microcontroladoras arduino WiFi en cada habitación al que se le conectarán pulsadores que se podrán activar y mandarán su señal a un microcontrolador Ethernet que se ubicará en cada planta del edificio. Todos los micros WiFi tendrán conexión con el micro Ethernet que actúa como servidor. Además, los profesionales de la residencia podrán acceder a un servidor web para recibir la información y poder generar una señal de vuelta que los pacientes podrán recibir mediante señales luminosas de bajo consumo diferenciadas por colores específicos en función del estado de atención de la solicitud.
La complejidad del proyecto estará determinada por la gestión conjunta en tiempo real de todas las comunicaciones de información entre clientes y servidor, además de también visionar la información en un servidor web y poder mandar una respuesta que llegue al usuario.
Para hacerse una idea gráficamente de cómo sería el proyecto, se puede observar el siguiente esquema:

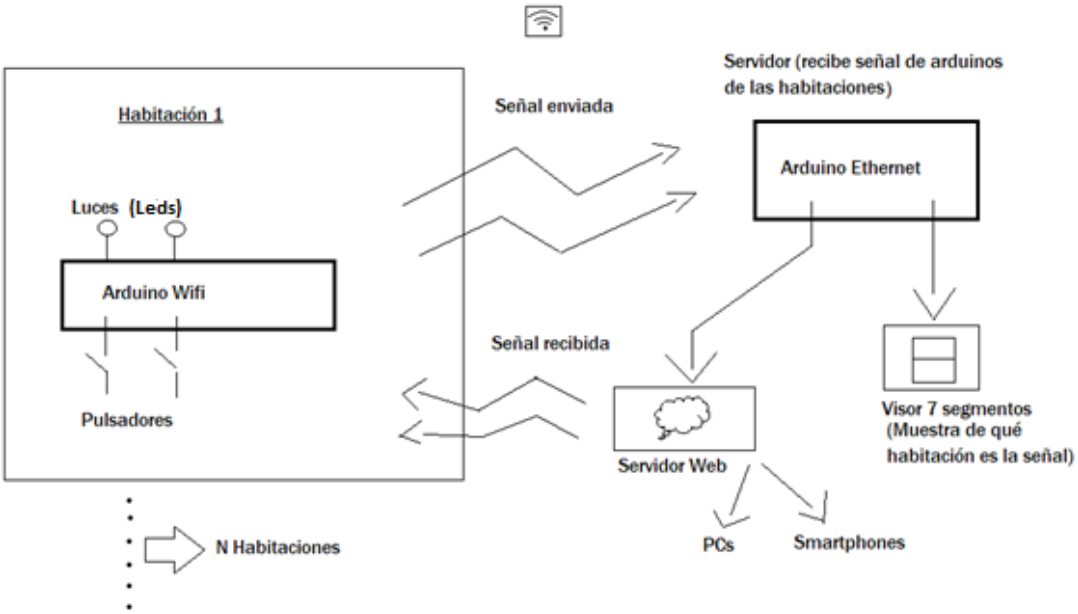


Ilustración 20: Esquema del proyecto

Como se puede observar en el esquema también se añadiría un display de 7 segmentos conectado al servidor para poder visualizar directamente la

habitación que ha mandado su señal. Desde el Servidor web los profesionales tienen la posibilidad de mandar una señal de que se ha recibido la llamada que recibirá el paciente mediante la señal luminosa correspondiente.
Para la realización de esto proyecto se ha diseñado un prototipo con una placa Arduino Ethernet como servidor y 2 placas Arduinos WiFi MKR1000 como clientes para simular el funcionamiento del sistema. Aunque en la práctica se usarían tantos clientes como habitaciones tuviera el edificio.

4.2 Montaje

Vamos a realizar tres montajes distintos para nuestro prototipo del proyecto. Uno será el servidor con el Arduino Ethernet y los otros dos serán los clientes, que en la práctica pueden ser muchos más, pero para nuestro prototipo con tener dos nos vale.

4.2.1 Montaje del Servidor:

En primer lugar, el esquema del montaje del servidor se puede ver en la siguiente imagen:

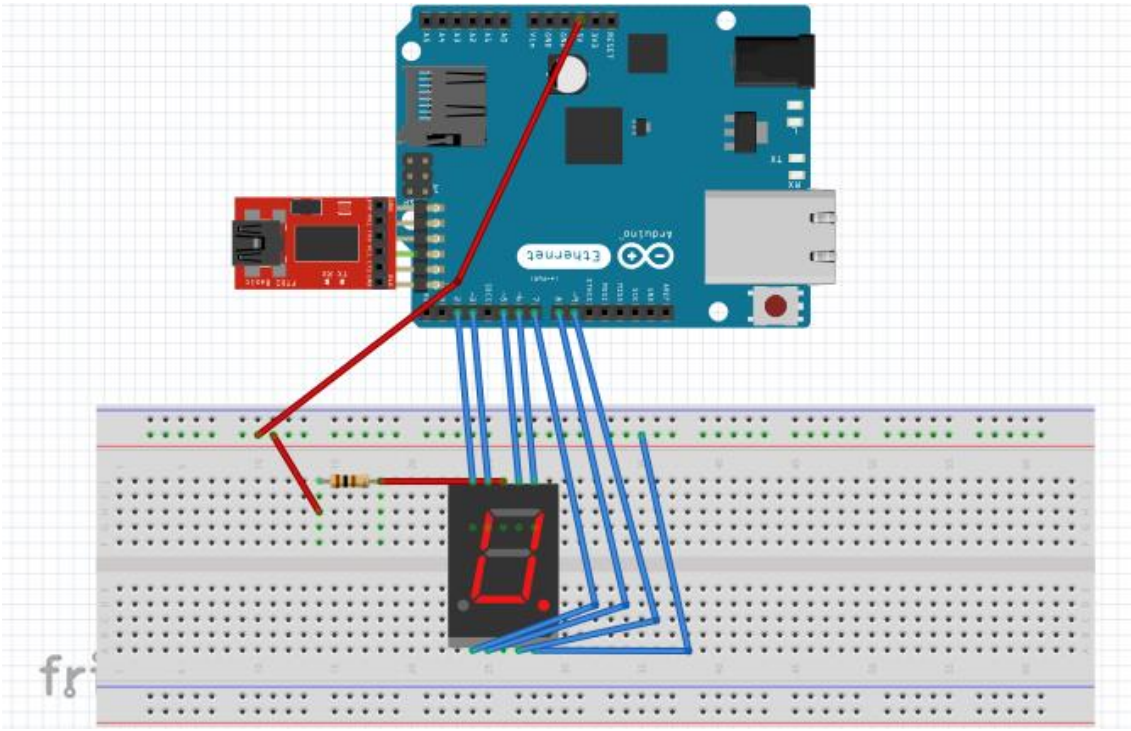


Ilustración 21: Montaje del Servidor

En este montaje se han incluido los siguientes componentes:

- Arduino Ethernet.
- Display de 7 segmentos.
- Resistencia de 100 Ohm.
- Cables.
- Protoboard.
- Módulo FT232RL FTDI.
- Cable Ethernet (no aparece en el esquema).
- Cable mini USB (no aparece en el esquema).

Como ya se ha explicado en capítulos anteriores, el display de 7 segmentos tiene cada pin asociado a un led que se iluminará en el caso de que se active a nivel bajo en nuestro caso. Por lo que los pines del display están conectados con 7 salidas digitales de nuestro microcontrolador, además de la alimentación que, como se puede observar será de 5V y se ha colocado una pequeña resistencia de 100Ohm para limitar la corriente que le llega y evitar sobrecalentamientos.

Además, como también se comentó anteriormente, se ha usado un módulo adaptador convertidor llamado FT232RL FTDI. Este módulo es un convertidor de RS232 a USB, a través de él se puede alimentar a 5V la placa y además debido a su conector miniUSB se puede conectar con el ordenador para subir el programa a la placa.

4.2.2 Montaje del cliente

El prototipo se ha realizado con el diseño y montaje de 2 clientes pero en la práctica se podrían introducir más clientes, los necesarios que se presenten en el proyecto en cuestión.

El montaje de los clientes es el mismo en todos ellos, lo único que cambiará serán aspectos en la programación de cada uno de ellos, cuestión que se abordará más adelante.

El esquema del montaje de los clientes es el siguiente:

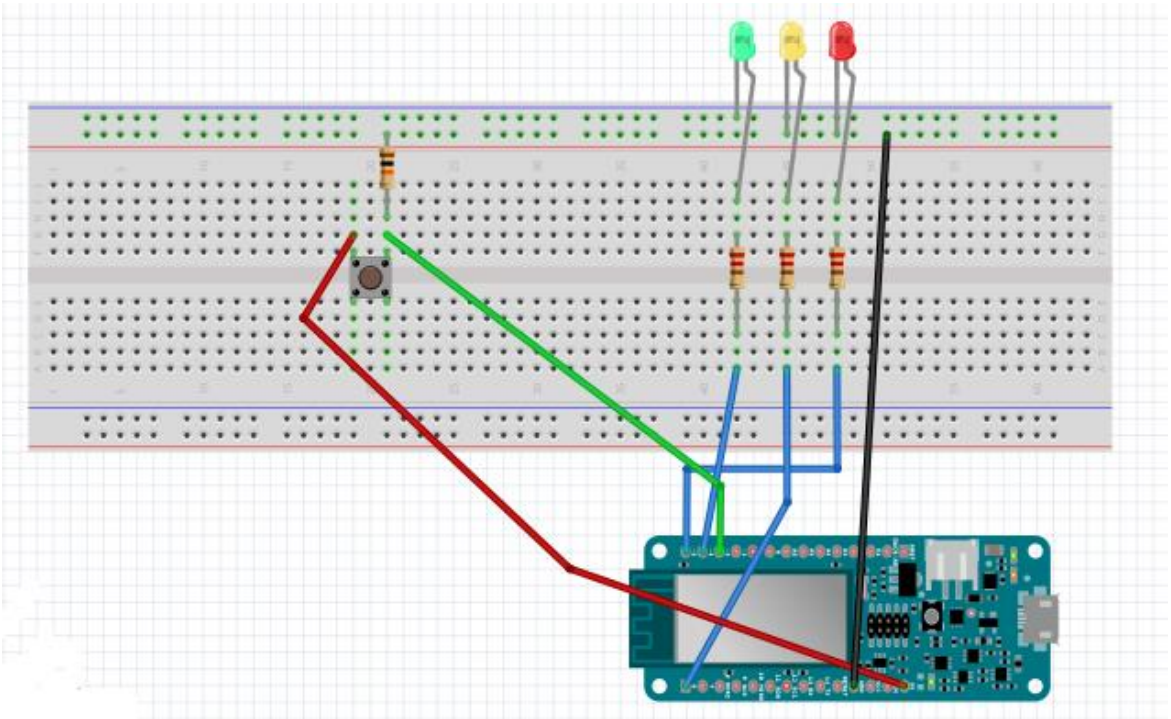


Ilustración 22: Montaje del Cliente

En este diseño se han requerido los siguientes componentes:

- Arduino WiFi MKR1000
- 3 resistencias de 220Ohm y 1 de 10k Ohm.
- Pulsador
- 3 LEDs de alta luminosidad de distintos colores
- Cables
- Cable micro-USB.
- Protoboard.

El Pulsador se tiene que conectar a una entrada digital que leerá el valor (0 o 1) dependiendo si está pulsado o no. Pero hay que tener en cuenta que para evitar el ruido en el circuito y poder establecer un 0 lógico en el estado de reposo, hay que introducir una resistencia de pull down (en nuestro caso es de pull down porque queremos un 0 en reposo, si quisiéramos un 1 en reposo sería de pull-up).

En la siguiente imagen se observa cómo queda el circuito en el caso de pull up y pull down.

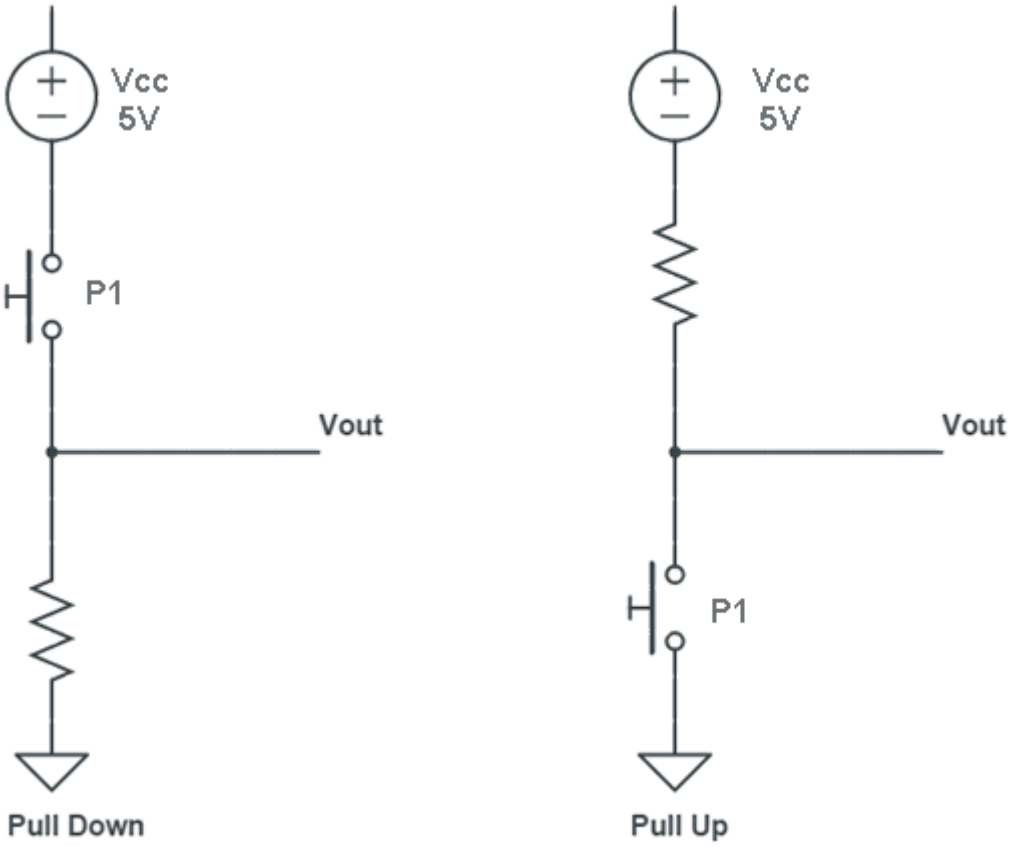


Ilustración 23: Resistencia pull down/pull up

En la configuración pull down, cuando el circuito está en reposo como se muestra en la imagen de arriba, la caída de tensión en la resistencia es prácticamente 0V (LOW), en cambio si pulsamos P1, dejará pasar la corriente y tendremos una diferencia de potencial de 5V **(**HIGH**)**. Este es el uso normal del estado LOW y HIGH. La resistencia empleada es de 10k Ohm pero puede cambiar su valor. Se suele usar una resistencia al menos 10 veces menor a la impedancia del PIN digital, los pines digitales que funcionan como entrada, tienen una impedancia variable, en función de la tecnología que se use en el chip, el fabricante, etc… pero es muy común que esta impedancia se encuentre en torno a 1MΩ. En nuestro circuito alimentado a 5V, si tenemos una resistencia de pull down de 10KΩ, la corriente que va a entrar en el circuito es de 0.5mA, algo que en términos de consumo es despreciable, ya que supone una potencia de 2.5mW.

Respecto a los LEDs, debido a que tienen polaridad, en primer lugar los tenemos que colocar con el cátodo a masa y el ánodo a la resistencia, que luego se va a polarizar con los pines de salida de la placa.

Además, incluso conectando el dispositivo con la polaridad correcta, con una tensión muy baja la corriente es despreciable y no es suficiente para verlo iluminado. Esto ocurre cuando se alcanza un cierto valor de tensión que llamamos tensión de polarización directa (Vd), que depende del tipo de diodo.

COLOR DEL LED	TENSIÓN UMBRAL
Rojo	1,6V
Rojo alta luminosidad	1,9V
Amarillo	1,7V a 2V
Verde	2,4V
Naranja	2,4V
Blanco brillante	3,4V
Azul	3,4V
Azul 430nm	4,6V

Ilustración 24: Tensiones umbral de los diodos LED

En esta imagen observamos la tensión umbral de cada tipo de LED en función de su color.

Por lo que tenemos que calcular el valor correcto de la resistencia de la siguiente forma:

$$Resistencia(\Omega) = \frac{Tensión\ de\ alimentación(v) - Tensión\ umbral\ LED}{Corriente\ admisible\ del\ LED}$$

$$Resistencia(\Omega) = \frac{5v-2v}{15mA} = 200\ \Omega$$

El valor de la corriente de Arduino suele ser de unos 15mA y de máxima unos 20mA por lo que se ha elegido 15mA para realizar el cálculo y, además, la tensión umbral de los tres diodos es de alrededor de 2V por lo que la resistencia teórica que se ha de utilizar es de 200 Ohm. En nuestro circuito se emplearán 3 resistencias de 220 Ohm cada una.

Se pensó en utilizar 1 diodo RGB en vez de tres distintos, pero no es posible debido a que nuestro microcontrolador Arduino WiFi MKR1000 solo dispone de dos pines con PWM y para formar todos los colores en el diodo RGB se necesitan 3 pines con PWM para ir variando el ciclo de trabajo y así ir creando los distintos colores.

4.3 Instalación e inicio del IDE de Arduino

Lo que hay que hacer en primer lugar es descargarse el entorno de desarrollo de la página oficial de Arduino.

https://www.arduino.cc/en/main/software

La versión más reciente hasta la fecha es la Arduino 1.8.8

En este caso se eligió la versión para Windows 64bits.

Una vez descargado el instalador se procede a la instalación del .exe y el proceso comenzará.

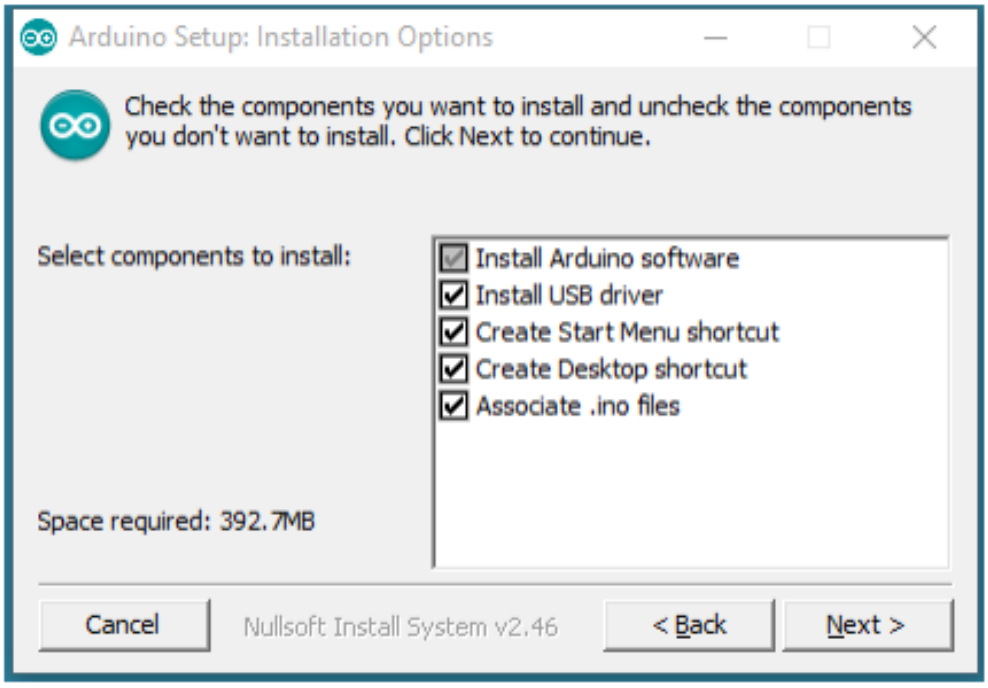


Ilustración 25: Instalación IDE Arduino

Seleccionamos todos los componentes a la hora de la instalación.

A continuación, hay que seleccionar la carpeta de instalación y luego esperar a que finalice el proceso.

Una vez finalizada la instalación del IDE desde dónde se programará la placa ya podemos conectarla a nuestro PC a través de un cable USB. Este cable además de servir para comunicar el ordenador con el Arduino también sirve para alimentarlo, por lo que no será necesario conectar una fuente de tensión al jack específico. Una vez conectado se encenderá el led de Power en la placa, lo cual quiere decir que tiene alimentación. La primera vez que se conecta el Arduino al ordenador, Windows lo detectará e instalará sus drivers automáticamente, a continuación, ya estamos listos para entrar en el IDE de Arduino. Se abrirá una ventana dónde automáticamente aparece un sketch para que empecemos a escribir código.

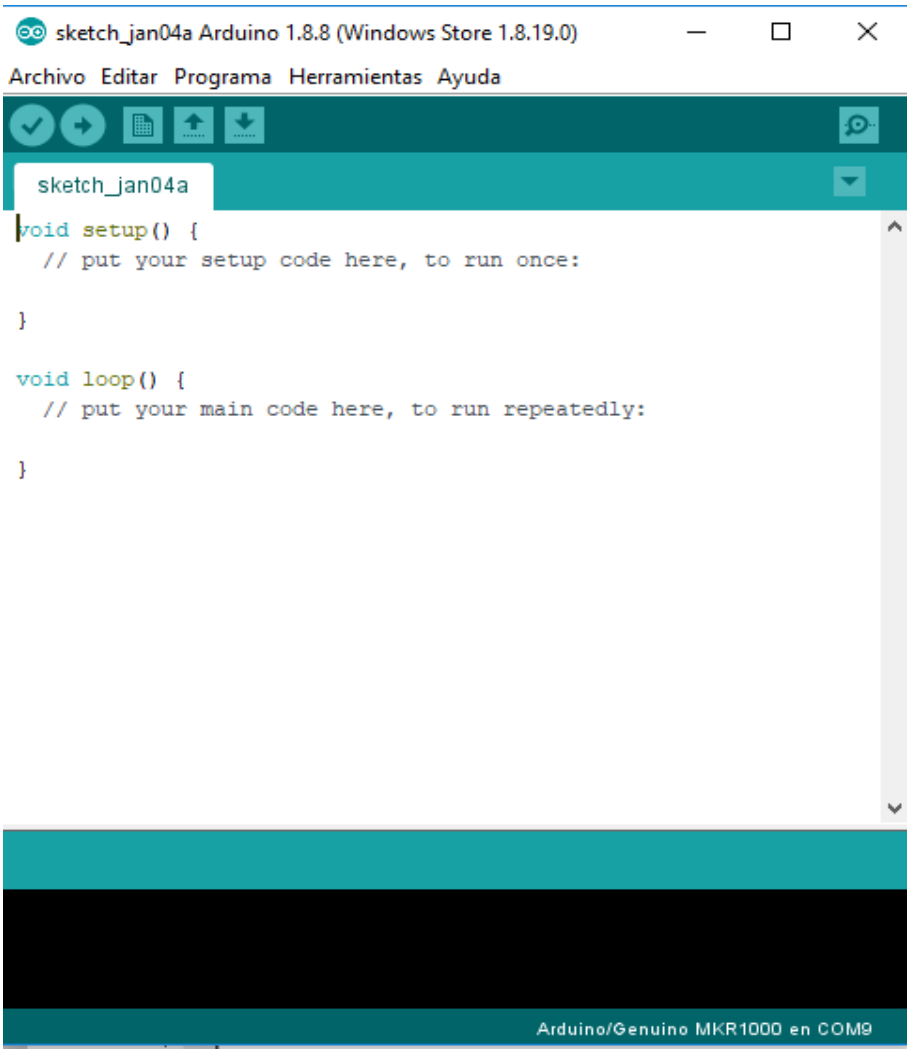


Ilustración 26: Sketch IDE Arduino

En primer lugar tenemos que configurar Arduino IDE para la placa Ethernet haciendo clic en herramientas y seleccionando Arduino Ethernet en el apartado de placa. Además, hay que señalar el puerto de nuestro ordenador al que lo tenemos conectado para que luego al compilarlo y subirlo a la placa nos lo reconozca sin problemas. El

puerto se puede comprobar en en el administrador de dispositivos de nuestro ordenador.

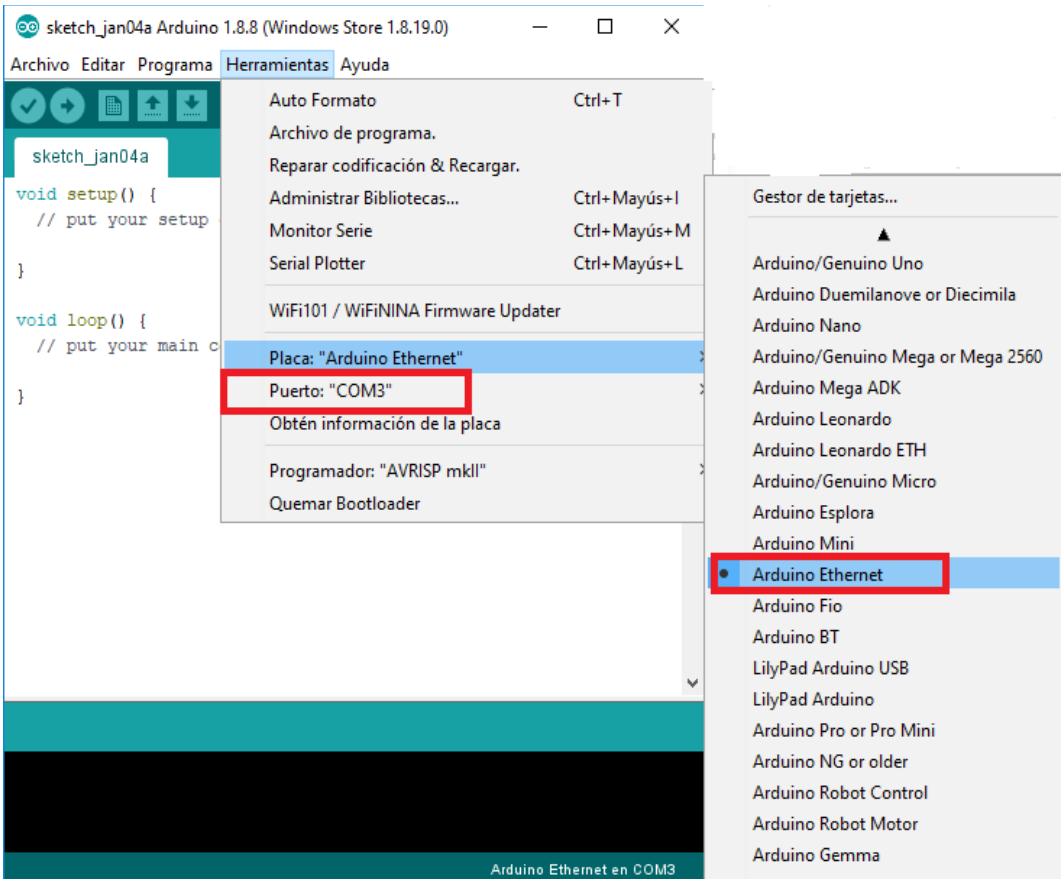


Ilustración 27: Puerto Arduino Ethernet

En el caso del Arduino WiFi MKR1000 hay que hacer lo mismo, pero este caso es algo más complicado debido a que en el Arduino IDE no viene directamente para añadirlo sino que hay que descargárlo desde el gestor de tarjetas del Arduino IDE.

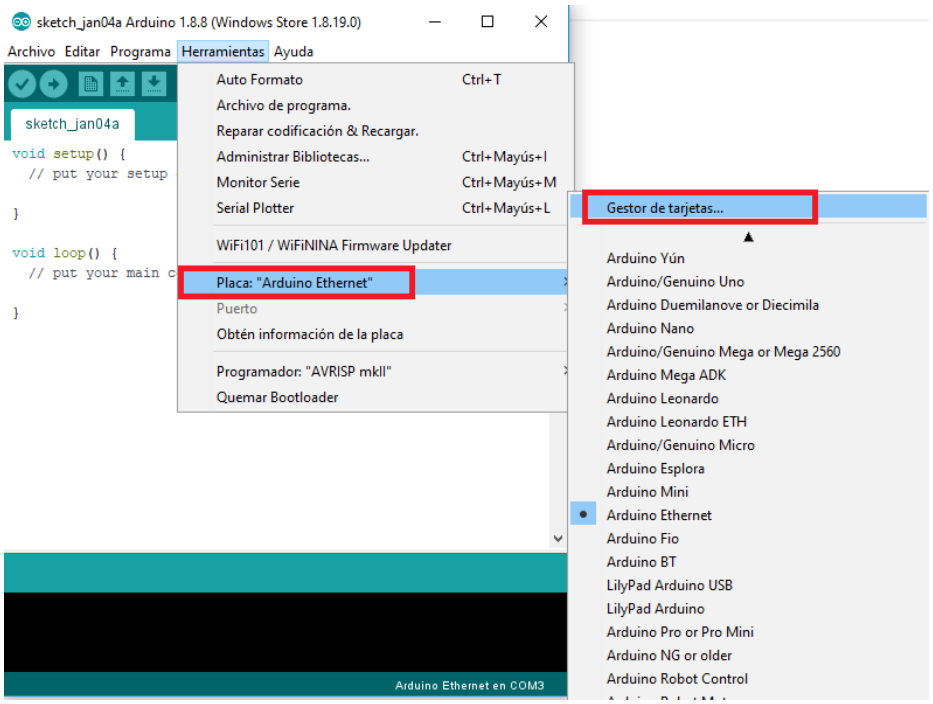


Ilustración 28: Puerto Arduino WiFi

A continuación, dentro del gestor de tarjetas hay que buscar la placa (Arduino WiFi MKR1000) y seleccionar la última versión e instalarlo.

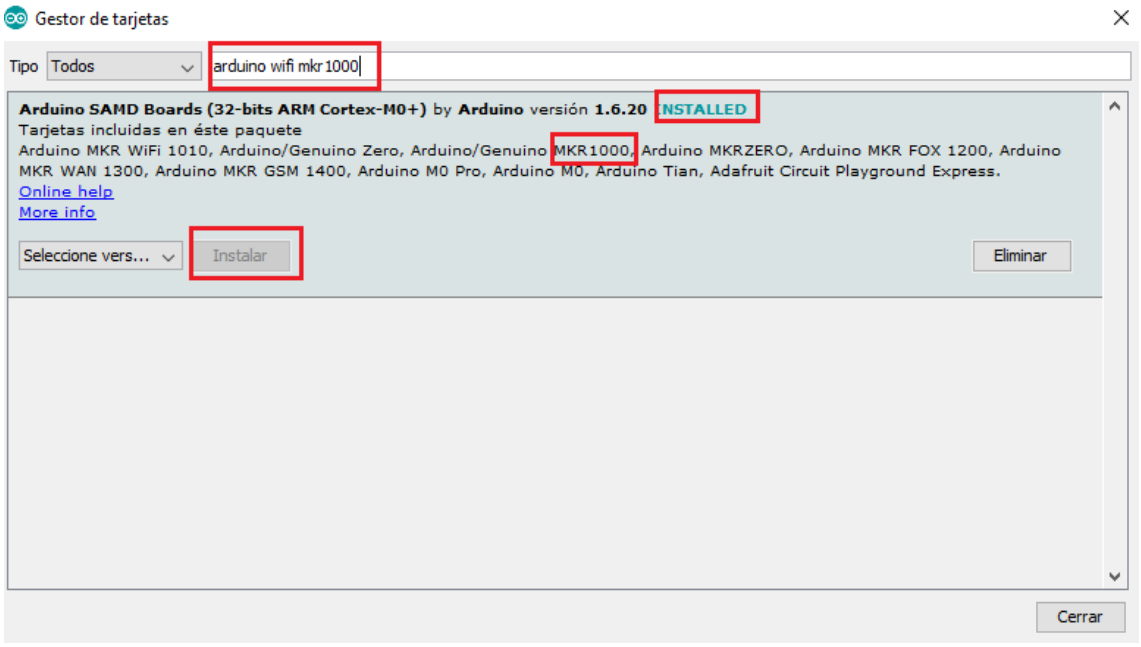


Ilustración 29: Gestor tarjetas, Arduino WiFi

En este caso ya indica que está instalado porque se hizo anteriormente para realizar el proyecto.

Una vez hagamos esto nos aparecerá en el apartado de “placas” todas estos arduinos que acabamos de instalar incluida la MKR1000.

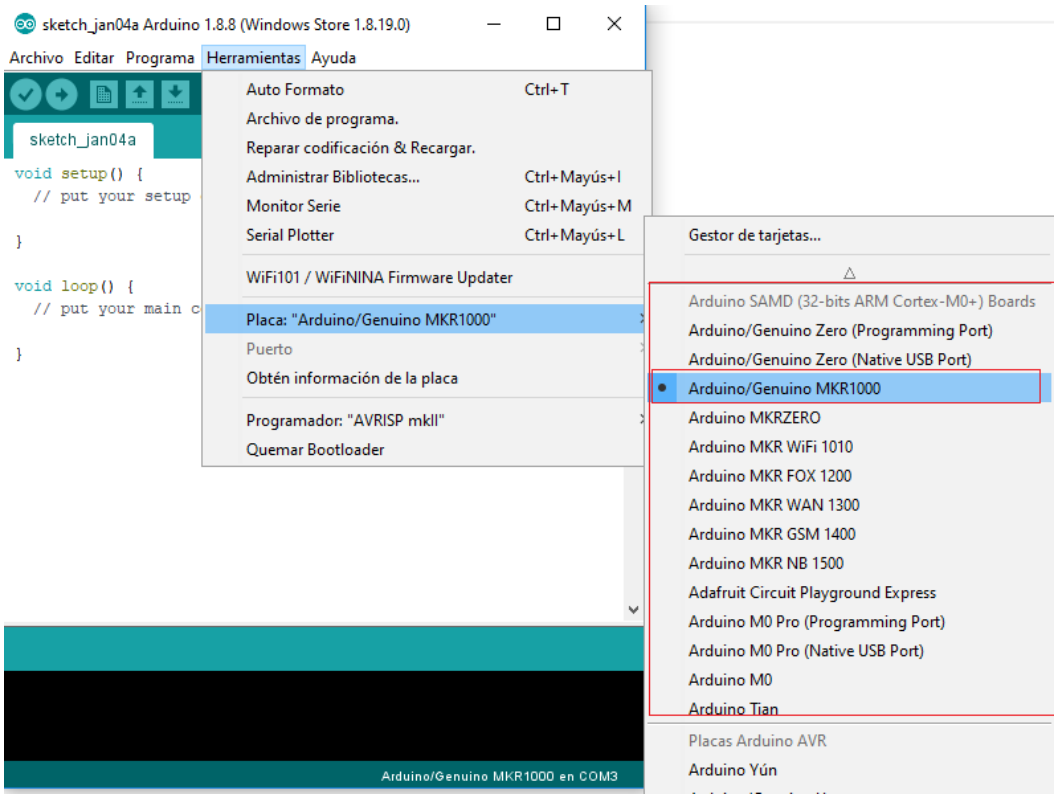


Ilustración 30: Puerto Arduino WiFi

Por último, para poder ver distintos aspectos de la programación tenemos la ventana de “Monitor Serial”, que una vez seleccionado el puerto correcto, cuando esté corriendo el programa en la placa, podremos observar en esta ventana lo que se desee ver.

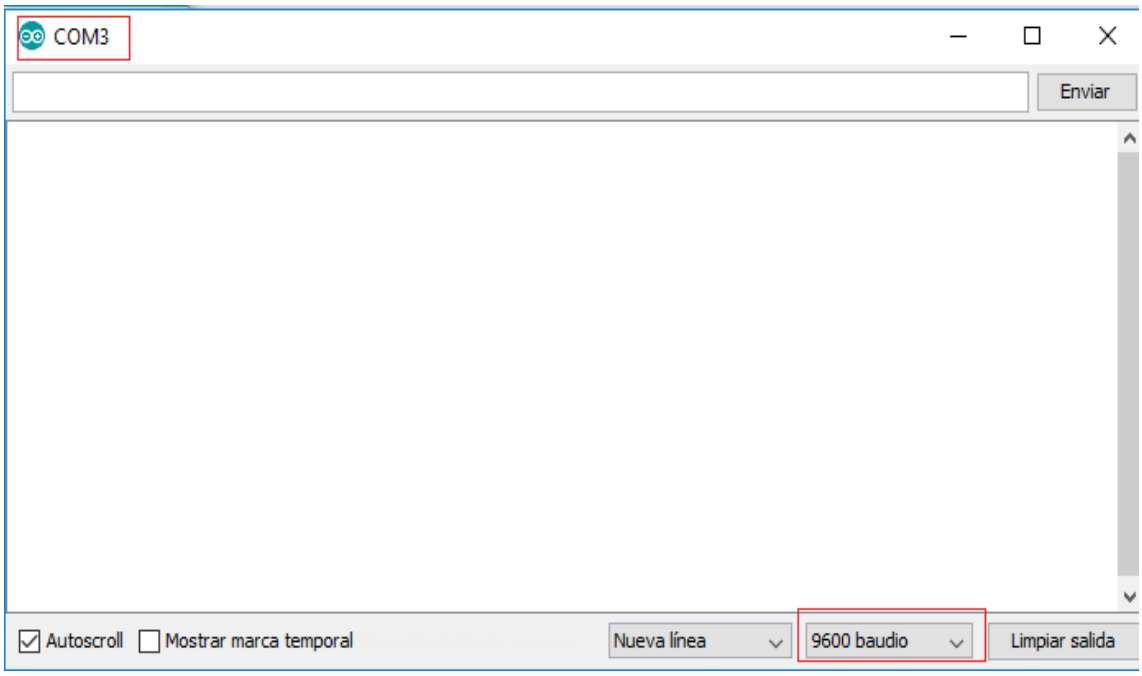


Ilustración 31: Monitor serial

Los Baudios son la velocidad de comunicación del ordenador con la placa, que normalmente se configura a 9600, aunque otras velocidades puedes ser soportadas. Este valor deberá ser configurado en el programa en la parte del "Void setup" mediante el comando *Serial.begin(9600)*.

4.4 Programación

A continuación, se va a comentar las partes más relevantes de la programación llevada a cabo tanto en los clientes como en el servidor.

4.4.1 Librerías

En primer lugar, tenemos que incluir las librerías que hemos tenido que utilizar en el Arduino WiFi MKR1000, nuestro cliente, y en el Arduino Ethernet, el servidor. En nuestro cliente hemos empleado: <WiFi101.h> y <SPI.h>

```
#include <SPI.h>
#include <WiFi101.h>
```

En el servidor hemos utilizado: <Ethernet.h> y <SPI.h>

```
#include <Ethernet.h>
#include <SPI.h>
```

- (Arduino P. O., 2016) La Librería WiFi101 permite utilizar el Arduino WiFi Shield 101 y la placa MKR1000. En nuestro caso solo nos interesa el MKR10000. Estas librerías conectan el Arduino a Internet de forma inalámbrica. Conectar la placa a una red WiFi es simple, no se requiere ninguna configuración adicional además del SSID y la contraseña. Puede servir como un servidor que acepta conexiones entrantes o un cliente que realiza conexiones salientes. La biblioteca admite el cifrado personal WEP y WPA2.
  La placa conectada al escudo se comunica con el escudo WiFi 101 mediante el bus SPI. Esto está en los pines digitales 11, 12 y 13 en el Arduino Uno y los pines 50, 51 y 52 en el Arduino Mega. En ambas placas, el pin 10 se utiliza como SS (Slave Select). El pin digital 7 se usa como un pin de saludo entre el escudo WiFi 101 y la placa subyacente, y no debe usarse. La biblioteca WiFi101es muy similar a Ethernet y a la biblioteca WiFi , y muchas de las llamadas de función son iguales.

- (Arduino P. O., Ethernet library, 2018) La librería Ethernet.h está diseñada para funcionar con Arduino Ethernet Shield, Arduino Ethernet Shield 2, Leonardo Ethernet y cualquier otro dispositivo basado en W5100 / W5200 / W5500. La librería permite que una placa Arduino se conecte a Internet. La placa puede servir como un servidor que acepta conexiones entrantes o un cliente que realiza conexiones salientes. La biblioteca admite hasta

ocho (W5100 y los tableros con <= 2 kB SRAM están limitados a cuatro) conexiones simultáneas (entrantes, salientes o una combinación).
La placa Arduino se comunica con el escudo utilizando el bus SPI. Esto está en los pines digitales 11, 12 y 13 en el Arduino Uno y los pines 50, 51 y 52 en el Arduino Mega. En ambas placas, el pin 10 se utiliza como SS.

- La librería SPI.h se ha incluido en ambos Arduinos.

  Esta biblioteca permite comunicarse con dispositivos SPI, con el Arduino como dispositivo maestro.

  La Interfaz Periférica Serial (SPI) es un protocolo de datos en serie síncrono utilizado por los microcontroladores para comunicarse con uno o más dispositivos periféricos rápidamente en distancias cortas. También se puede utilizar para la comunicación entre dos microcontroladores.

  Con una conexión SPI siempre hay un dispositivo maestro (generalmente un microcontrolador) que controla los dispositivos periféricos. Típicamente hay tres líneas comunes a todos los dispositivos:

  MISO (Master In Slave Out): La línea Slave para enviar datos al maestro.
  MOSI (Master Out Slave In): La línea Master para enviar datos a los periféricos.
  SCK (Reloj de serie): los pulsos de reloj que sincronizan la transmisión de datos generada por el maestro.

  Y una línea específica para cada dispositivo:

  SS (Selección de esclavo): el pin en cada dispositivo que el maestro puede usar para habilitar y deshabilitar dispositivos específicos.
  Cuando el pin de selección de esclavo de un dispositivo es un nivel bajo, se comunica con el maestro. Cuando está alto, ignora al maestro. Esto le permite tener múltiples dispositivos SPI compartiendo las mismas líneas MISO, MOSI y CLK.
  (Arduino P. O., SPI library, 2018)

4.4.2 Programación de los Clientes

A continuación, se va a comentar las partes más relevantes de la programación del cliente.

Lo primero es establecer la conexión WiFi con nuestra red.

Para ello declaramos las siguientes variables:

```
char ssid[] = "MOVISTAR_7C58";        // your network SSID (name)
char pass[] = "ECB2996D867E70DBBCC6"; // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0;  // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
```

En el SSid ponemos nuestra red WiFi y el pass ponemos la contraseña de la misma. El keyIndex no es necesario ponerlo en este caso porque se usa una red con WPA.

La función que hace conectarse a la red WiFi se llama “PrintWiFiStatus” y lo que hace resumidamente es que gracias a la librería WiFi101.h se puede conectar a la SSID que hemos puesto antes y además podemos imprimir en pantalla la IP que se nos ha asignado al conectarnos. Además, mediante el comando WiFi.RSSI() podemos saber el ancho de banda de nuestra señal. Una vez conectada a la red WiFi se encederá el LED rojo.

```
void printWiFiStatus() {
      // check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
  Serial.println("WiFi shield not present");
  while (true);       // don't continue
}

// attempt to connect to WiFi network:
while ( status != WL_CONNECTED) {
  Serial.print("Attempting to connect to Network named: ");
  Serial.println(ssid);                   // print the network name (SSID);

  // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
  status = WiFi.begin(ssid, pass);
  // wait 10 seconds for connection:
  delay(10000);
}
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());

// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);

// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");


  Serial.print(rssi);
  Serial.println(" dBm");

digitalWrite(ROJO, HIGH);

}
```

Otra parte importante sería cómo se ha establecido la conexión del servidor. Para ello hay que establecer la dirección IP de nuestro servidor y el puerto de enlace.

```
// Informacion de nuestro servidor
const char host[] = "192.168.1.37"; //Direccion IP de nuestro servidor
const int httpPort = 80; //Puerto de enlace
```

Si cambiase la dirección de nuestro servidor tendríamos que cambiar el valor del HOST.

Y en el Loop hay que establecer la conexión con el servidor de esta forma:

```
  // Se intenta la conexion con el servidor
  WiFiClient client;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }
```

Si no se pudiera conectar con el servidor aparecería un mensaje en el monitor serial de "conexión fallida" y seguiría en ese bucle hasta que se lograse conectar.

Además, una vez conectado con el servidor, si el cliente no se encuentra disponible durante 5 segundos se para y se sale de la conexión con el servidor. En cambio, si el cliente se encuentra disponible recibe la información del servidor de vuelta.

```
    unsigned long timeout = millis();
  while (client.available() == 0) {
    if (millis() - timeout > 5000) {
      Serial.println(">>> Client Timeout !");
      client.stop();
      return;
    }
  }


    while(client.available()){
     line = client.readStringUntil('\r');
    Serial.println(line);
    if (line == "recibidol"){
  digitalWrite(VERDE, HIGH);
  digitalWrite(ROJO, LOW);
  digitalWrite(AMARILLO, LOW);
    }
  Serial.println("enciendo el led");
}
```

Como se puede observar en el código si se recibe del servidor “recibido1” sabrá que se ha pulsado el botón de recibido del servidor web y puede cambiar de color el led del cliente. Esta es una de las partes que cambia en los clientes ya que en el cliente 2 se escribirá “recibido2” en el caso de que se pulse el botón en el servidor web del cliente 2.

Por último para mandarle la señal de que el pulsador ha sido pulsado al servidor le mandamos al server una URL con el valor del pulsador (0 o 1) y para que reconozca que es un cliente u otro se ha utilizado la MAC de cada una de las placas clientes.

```
String macID= "D0A3"; //String donde se almacenara nuestra macID
```

En macID se guarda la mac de nuestro cliente, en el caso del cliente 2 tendrá otro nombre distinto para reconocerlo.

```
String url = macID;
url += "pulsador=";
url += pulsador;

Serial.print("Requesting URL: ");
Serial.println(url);

 // Se envia la solicitud al servidor
client.print(String("GET /") + url + "/ HTTP/1.1\r\n");
```

Mediante el *client.print* le mandamos al servidor la solicitud con el valor del pulsador y la MAC para que pueda saber de qué cliente se trata. Luego el servidor dependiendo de qué MAC lea puede ejecutar unas sentencias u otras.

4.4.3 Servidor. Código Arduino

Lo primero que hay que hacer es configurar el Arduino Ethernet como servidor y además conectarlo a Internet. Este proceso es más sencillo que el del Arduino WiFi ya que la conexión a Internet se hace mediante cable Ethernet.
En el apartado de las variables se configura como servidor:
EthernetServer server(80);
Entre paréntesis se pone el puerto que usamos.

En el Setup realizamos la configuración definitiva para conectarnos a la red.

```
void setup() {
  config_salidas ();


  // start the serial library:
  Serial.begin(9600);
  // start the Ethernet connection:
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    // no point in carrying on, so do nothing forevermore:
    for(;;)
      ;
  }
  // print your local IP address:
  Serial.println(Ethernet.localIP());
  server.begin();
}
```

Anteriormente se ha puesto la MAC de nuestro dispositivo en la declaración de las variables, y con ella y "Ethernet.Begin" se comienza a conectar a la red. Se le asignará automáticamente una IP que luego será está IP a la que se tienen que conectar los clientes. También se puede asignar manualmente una IP al servidor, pero se ha preferido no hacerlo así debido a que puede entrar en conflicto con otros dispositivos que estén conectados a la red.

Se comenzará a comportar como servidor con el comando "Server.begin()". También se puede observar en el Setup la función config_salidas () donde se configuran todos los pines de salida necesarios en el programa, sobre todo los del display.

La parte en la que el servidor está pendiente de recibir a los clientes es la siguiente:

```
  // Si el cliente esta conectado
  EthernetClient client = server.available();
  if (!client) {
 Serial.println("no hay cliente");
    return;
  }

  // Se realiza la lectura de la solicitud
  String req = client.readStringUntil('\r');
  Serial.println(req);
```

Si el servidor está disponible y hay un cliente que pide conectarse con el mismo se recoge en la variable 'req' lo que le mande el cliente.

La diferenciación entre cliente 1 y 2 se realiza con las siguientes sentencias:

```
if (req.indexOf("/D0A3") != -1){  //Dispositivo 1;


 if (req.indexOf("/ECF0") != -1){  //Dispositivo 2;
```

Como se puede observar dependiendo del valor de la variable 'req' que ha sido leída anteriormente, si la variable muestra el valor de la MAC del dispositivo 1 se trata del cliente 1, sino será el cliente 2.

A partir de ese punto, ya identificado y conectado con cada cliente, podemos poner el número en el display dependiendo si es el cliente 1 ó 2. Además, se decodifica el valor del pulsador para saber si es 0 ó 1 en cada momento:

```
if (req.indexOf("/D0A3") != -1){  //Dispositivo 1; escribe el ID del dispositivo

  String value1;
  val = 0;
  int Start1 = req.indexOf("=");
  int Finish1 = req.indexOf('/', Start1 + 1);
  Serial.print("start1: ");
  Serial.println(Start1);
  Serial.print("Finish1: ");
  Serial.println(Finish1);
  value1 = "";

  for (int i = Start1 + 1; i < Finish1; i++)
{
  value1 = value1 + req.charAt(i);
}

 pulsador1=value1.toInt();

  Serial.print("el pulsador1 tiene el valor: ");
  Serial.println(pulsador1);
```

Como se puede observar se analiza la variable 'req' para identificar el valor del pulsador. El valor del pulsador1 y pulsador2 (en el caso del cliente2) se utilizarán en el código HTML para mostrarlo en la página web.

Luego se evaluará la variable 'req' en los siguientes casos:

- Si se ha introducido en el navegador la IPserver/menú:
- Si una vez entrado en la página web se pulsa el botón1.
- Si una vez entrado en la página web se pulsa el botón 2.

En todos esos casos se le asigna a la variable 'val' un valor distinto para luego introducir el código HTML correspondiente para los determinados valores de esa variable.

```
else if (req.indexOf("/menu") != -1){
val = -2; // Se imprimira el menu de la pagina web
Serial.println("se ha dado al menu");
}
  else if (req.indexOf("/?var=ACTIVAR1") != -1){
val = -3; // Se imprimira aviso paciente1
a=1;
  }
else if (req.indexOf("/?var=ACTIVAR2") != -1){
val = -4; // Se imprimira aviso paciente2
a=2;
}
```

Por último mediante el tratamiento de unas variables usadas en el programa, se mandará la señal “recibido1” o “recibido2” a los clientes correspondientes mediante la función “client.println()” dependiendo si se ha pulsado el botón 1 o 2 en la página web.

```
if ((val>=0)&&(z==1)){

        s="recibido1";
        Serial.println("s1\r\n");
        client.println("recibido1");
        delay(1);
        z=0;


    }
    if ((val>=0)&&(z==2)){

        s="recibido2";
        Serial.println("s2\r\n");
        client.println("recibido2");
        delay(1);
        z=0;

    }
```

Además, al pulsar uno de eso botones, se apagará el display que lucía el número del cliente que se había conectado:

En el caso de que se pulse para el cliente 1:

```
 if (Pulsador1==1){
  digitalWrite(6, LOW);
  digitalWrite(9, LOW);
 }
     else if ((Pulsador2==0) && (Pulsador1==0) && (a==1)){//hasta que no se da la señal desde el serv que se ha recibido no se apaga el display
digitalWrite(8, HIGH);
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(9, HIGH);
digitalWrite(5, HIGH);
digitalWrite(6, HIGH);
digitalWrite(7, HIGH);
a=0;
z=1;
    }

}
```

En el caso de que se pulse para el cliente 2:

```
    if (Pulsador2==1){
  digitalWrite(7, LOW);
  digitalWrite(2, LOW);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
  digitalWrite(8, LOW);
    }
    else if ((Pulsador2==0)&&(Pulsador1==0) && (a==2)){//hasta que no se da la señal desde el serv que se ha recibido no se apaga el display
digitalWrite(8, HIGH);
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(9, HIGH);
digitalWrite(5, HIGH);
digitalWrite(6, HIGH);
digitalWrite(7, HIGH);
a=0;
z=2;
```

4.4.3.1 Servidor. Código HTML

En este apartado se va a poder observar el código utilizado para la realización de la página web. Para ello se maneja el uso de la variable 'val' para los distintos casos que se requieren:

En el caso de que val < - 1:

```
else if (val < -1){
  Serial.println("Iniciamos un header comun");
  // Si nuestro clente es otro dispositivo

  // Se prepara la respuesta. Iniciamos con un header comun:
  client.println("HTTP/1.1 200 OK\r\n");


   client.println("<!DOCTYPE HTML>\r\n<html>\r\n");
```

Se va a iniciar una cabecera común de la página web requerida para su funcionamiento.

En el caso de que val ==-2:

```
if (val == -2) {

  //Serial.println("menu para elegir dispositivo");
    // Se imprimen en una pagina web el menu para elegir que dispositivo se desea realizar la lectura
    client.println("<head><title>Arduino Ethernet WEB SERVER .</title></head>");
    client.println("<body><center><br><br><br><h1><u> P&aacute;gina Web que refleja los dispositivos activados </u></h1>");
     client.println("<form action=\"192.168.1.37/\" method=\"GET\">");
    client.println("<meta http-equiv=\"refresh\" content=\"1\"></head>"); //Funcion que realiza el refrescamiento cada segundo
    client.println("<center><h2> Se puede mandar la se&ntilde;al de socorro al activarse los pulsadores</h2>");

    client.println("Pulsador1 = "  + String(Pulsador1) + "<br>");
    client.println("<input type=\"submit\" name=\"var\" value=\"ACTIVAR1\"> Dispositivo 1.<br>");


    client.println("<br>");

    client.println("Pulsador2 = "  + String(Pulsador2) + "<br>");
    client.println("<input type=\"submit\" name=\"var\" value=\"ACTIVAR2\"> Dispositivo 2.<br>");
```

En este caso entraría porque se ha introducido en el navegador el nombre de la IP del servido/ menú. En nuestro caso http://192.168.1.37/menu

Al introducir esto en el navegador se imprimirá el menú de la página web donde se observará el valor de los pulsadores en tiempo real, y se introducirá dos botones para pulsarlos en el caso de que se quiera mandar a los clientes una señal de recibido.

Además, la página web se actualizará cada segundo para poder ver el valor de los pulsadores en tiempo real:

client.println("<meta http-equiv=\"refresh\" content=\"1\"></head>");

En el caso de val==-3:

```
}
        if (val == -3) {

  // cliente1
  client.println("<head><title>Mediciones de dispositivo 1.</title>");
  client.println("<meta http-equiv=\"refresh\" content=\"1\"></head>");
  client.println("<center><h2> SE HA MANDADO EL AVISO AL PACIENTE 1</h2>");
  client.println("<p><em> Volver al menu principal.</em></p></body></center>");


}
```

Si val==-3 significa que se ha pulsado el botón de aviso al cliente 2 y, por lo tanto, saldrá un mensaje en la página web de que se ha mandado la señal al cliente correspondiente.

En el caso de val == -4:

```
}
if (val == -4) {

  // cliente 2
  client.println("<head><title>Mediciones de dispositivo 2.</title>");
  client.println("<meta http-equiv=\"refresh\" content=\"1\"></head>");
  client.println("<center><h2> SE HA MANDADO EL AVISO AL PACIENTE 2</h2>");

  client.println("<p><em> Volver al menu principal.</em></p></body></center>");
}
```

Si val==-4 significa que se ha pulsado el botón de aviso al cliente 2 y, por lo tanto, saldrá un mensaje en la página web de que se ha mandado la señal al cliente correspondiente.

En el caso de que el servidor web reciba otra petición desconocida se verá en la página web:

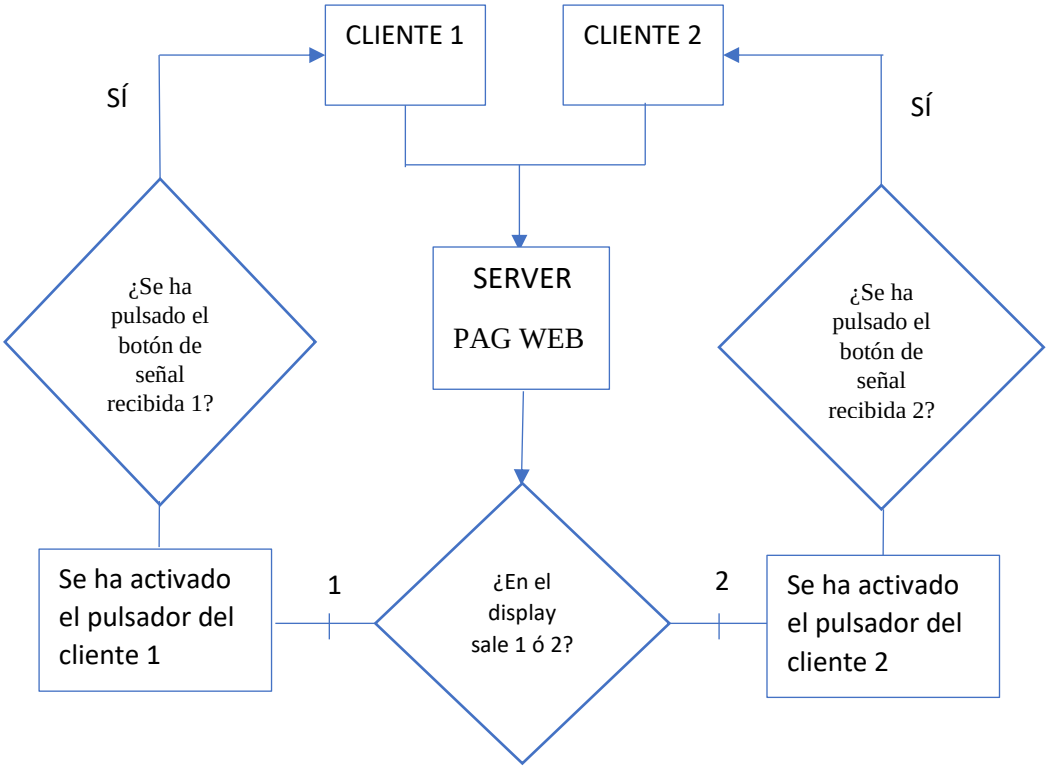
```
else
{
 client.println("Invalid Request.<br>");
}
```

5. RESULTADOS Y PRUEBAS

Todas las pruebas se han realizado observando y analizando el comportamiento en sí del sistema sabiendo lo que supuestamente estaba destinado a realizar. Además, en el monitor serial del IDE de Arduino mediante el comando Serial.Print() se puede ir observando el valor de las variables en cada parte del programa, así como de las peticiones del cliente para ver si sus valores eran los adecuados, y corregirlos en caso de que no fueran así. Esto fue fundamental a la hora de corregir los problemas que surgían a lo largo de la programación.

A continuación, se va a mostrar una serie de capturas acerca del sistema y de su funcionamiento en la práctica.

En primer lugar, para explicar de forma sencilla el funcionamiento del sistema diseñado se ha realizado el siguiente diagrama de flujo:



Como se puede observar en el flujograma, en primer lugar, la información acerca del estado de los pulsadores pasa de los clientes al servidor y se sube a la página web diseñada. En el servidor se puede saber si se ha activado el pulsador del cliente 1 ó 2 mediante el display, además de que en la página web se observa que el pulsador se vuelve a 1. Una vez que el

pulsador de cualquier cliente se ha activado ya, desde la página web, se puede devolver la información al cliente determinado mediante la activación del botón de 'Señal recibida' y le llegará esa información al cliente correspondiente dependiendo del botón que se haya activado.

A continuación, se ha realizado una prueba de concepto donde hemos comprobado que todo el sistema diseñado funciona correctamente. Habría que añadir que la aplicación web puede completarse en un futuro para añadir una mejor funcionalidad al sistema.

El menú de la página web al introducir la IP del servidor/ menú en navegador web tendría la siguiente interfaz:

Ilustración 32: Menú página web

Como se puede observar Aparece el valor de los pulsadores 1 y 2 en tiempo real, actualizándose cada segundo. Además, hay dos botones para poder mandar la señal de vuelta en caso de que se haya pulsado el pulsador.

Ilustración 33: Página web pulsador1 activado

Si el pulsador1 se mantiene pulsado se observará en el menú de la página web como Pulsador1 se pone con el valor 1.

Además, en el display que está conectado en el Arduino Ethernet se puede observar que cuando se pulsa el pulsador del cliente 1 se pone un 1 en el display

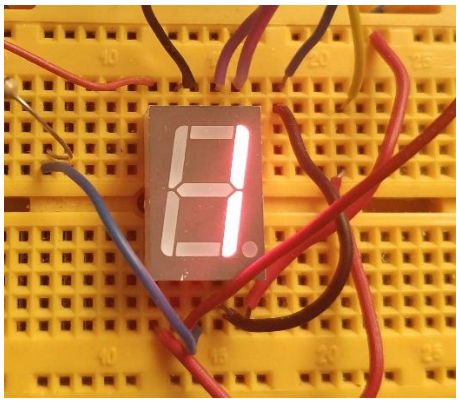


Ilustración 34: display con pulsador 1 activado

Y al activar el botón 'Activar1' aparecerá el siguiente mensaje:

Ilustración 35: Página web, aviso a cliente1

Además, en el display dejará de lucir el número 1.

En caso de que se mantenga pulsado el pulsador 2 se observará que el pulsador 2 tomará el valor de 1:

Ilustración 36: Página web:Pulsador2 activado

Y, como en el caso del pulsador1, en el display del servidor se observará como luce el número 2 con solo pulsar el pulsador 2 en el cliente 2.

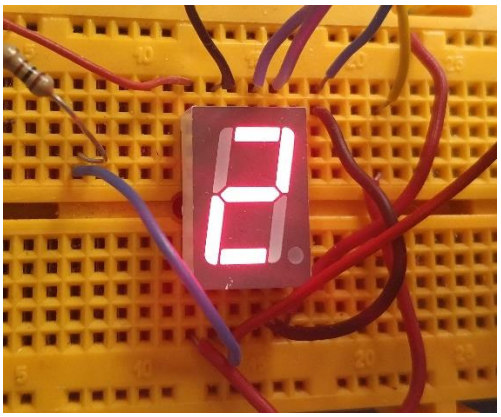


Ilustración 37: Display con pulsador 2 activado

Al igual que con el botón 'Activar1', al pulsar 'Activar2' se observará el siguiente mensaje en la web, además de que se mandará la señal al cliente 2 y dejará de lucir el display.

Ilustración 38: Página web, aviso a cliente2

Además a todo esto, en ambos clientes se han puesto 3 diodos Leds con los colores rojo, amarillo y verde.

Estos colores representan lo siguiente:

Cuando los clientes se conectan a la red WiFi correspondiente, se encenderá el Led rojo. A continuación, si se pulsa el Pulsador del cliente, se encenderá el Led amarillo, así sabrá el usuario que se ha pulsado el botón y que la señal ha llegado al servidor. Una vez que los profesionales ven en el servidor web que se ha pulsado el pulsador de uno de los clientes y le dan al botón de 'Activar' el Led amarillo se apagará y se encenderá un Led verde como señal de que la llamada de emergencia ha sido escuchada.

El sistema diseñado parece bastante eficiente debido a que, basados en las pruebas realizadas, el valor del pulsador se actualiza bastante rápido en el servidor web por lo que la información llega muy veloz a los profesionales y eso es muy importante a la hora de actuar en caso de emergencia. También se actualizan muy rápido en el display los números que lucen al pulsarse los pulsadores correspondientes en los clientes.

6. PLIEGO DE CONDICIONES

6.1 PLIEGO DE CONDICIONES GENERALES

Este Pliego, también llamado Pliego de Condiciones Técnicas, es un resumen de las características que tiene que cumplir el proyecto que se ha llevado a cabo.

El objetivo principal de este documento es el de diseñar y programar un sistema para la gestión de una red de sensores para la asistencia a personas dependientes, donde se va a plasmar toda la información y materiales que se ha necesitado para su realización, así como los resultados obtenidos.

El proyecto está compuesto por la siguiente documentación, que es de obligado cumplimiento:

- Memoria.
- Pliego de condiciones.
- Anexos.

6.2 PLIEGO DE CONDIONES PARTICULARES

Para la realización de este proyecto se ha necesitado la revisión de numerosa información acerca de redes inalámbricas, de programación en Arduino y de programación en HTML. Así como toda la documentación necesaria acerca de los dataheet de las placas microcontroladoras utilizadas en el proyecto.

Las herramientas software empleadas han sido las siguientes:

- IDE Arduino 1.8.8, que se puede descargar gratuitamente desde la página web de Arduino.
- Microsoft Office en su versión 365. En particular se ha empleado el programa Microsoft Word para la escritura de la memoria y Microsoft Power Point para la realización de la presentación del proyecto.

Las herramientas hardware usadas en el proyecto han sido:

- Pc portátil Asus K55V con las siguientes especificaciones:
  - 250 GB de disco duro sólido.
  - Windows 10 Home 64.
  - Procesador Intel Core i7 3630QM, 2.4 GHz
  - 8 GB de memoria RAM.
- 2 Arduinos MKR1000 y 1 Arduino Ethernet.

- Resistencias, Leds y Displays de 7 segmentos.
- Cables USB para la conexión al PC de los microcontroladores.
- Cable Ethernet y conexión a WiFi para las comunicaciones empleadas.

7. PRESUPUESTO

A continuación, se van a recoger los costes que han generado los materiales a la hora de realizar el proyecto además del coste de la mano de obra. Esto se denomina Presupuesto de Ejecución Material (PEM).

La siguiente tabla recoge el gasto en material empleado para el proyecto:

Material	**Unidades**	**Precio**	**Utilización/duración**	**Total**
PC	1	500€	5 meses/ 6 años	41.66€
Arduino Ethernet	1	20€	5 meses/ 1 año	10€
Arduino WiFi MKR1000	2	40€	5 meses/ 5 meses	80€
Módulo adaptador FT232	2	8€	5 meses/ 5 meses	16€
Diodos Led	6	0.5€	5 meses/ 5 meses	3€
Display	2	3€	5 meses/ 5 meses	6€
Resistencias	10	0.1€	5 meses/ 5 meses	1€
Pulsador	2	2€	5 meses/ 5 meses	4€
Cable USB	3	1€	5 meses/ 2 años	0.75€
Cable Ethernet	1	3€	5 meses / 2 años	0.75€
TOTAL				163,16€

Si a los materiales le sumamos las horas de la mano de obra empleada en la realización del proyecto además de la escritura de la memoria obtenemos los siguientes resultados:

Trabajo	**Tiempo**	**Precio/Hora**	**Total**
Ingeniero (Realización del proyecto)	200	25	5000 €
Escritura	80	16	1280 €
Total			6280 €

Si realizamos la suma de las dos tablas nos saldría un precio total de 6443,16 €.

Además del Presupuesto de Ejecución Material, hay que incluir el Presupuesto de ejecución por Contrata (PC), donde se incluye el PEM junto con los gastos Generales, el Beneficio Industrial y los Honorarios de dirección y redacción.

Concepto	**Valor**	**Total**
PEM	1	6443,16 €
Gasto Generales y Beneficio Industrial	13% PEM	837,6 €
Honorarios de redacción	6% PEM	386,59 €
Honorarios de dirección	6% PEM	386,59 €
Total		8053,93 €
Total + 21% IVA		9745,25 €

El presupuesto total del proyecto asciende a 9745,25€.

8. CONCLUSIONES Y LÍNEAS FUTURAS

8.1 CONCLUSIONES

Como se puede observar a lo largo de este proyecto se ha conseguido diseñar un sistema sencillo pero con una buena funcionalidad y bastante intuitivo que además es fácilmente ampliable con la incursión de otros actuadores y sensores debido a que la comunicación entre los clientes y servidor sería del mismo modo, solo habría que cambiar la forma de codificar los datos que se obtengan.

Concretamente, después del desarrollo del proyecto las conclusiones que se pueden sacar son las siguientes:

- Los microcontroladores Ethernet y WiFi de Arduino que hemos usado en el proyecto nos permiten realizar multitud de tareas, que en un principio pueden parecer complejas, de una forma relativamente más sencilla.

- Partiendo de un proyecto sencillo se pueden llegar a desarrollar aplicaciones muy interesantes.

- La tecnología IoT se encuentra en continuo desarrollo en la actualidad con vistas a que va a tener una gran importancia en el futuro próximo debido a la gran cantidad de aplicaciones que se están implementando gracias a los avances en las comunicaciones inalámbricas.

Desde el punto de vista académico las conclusiones que se obtienen después de la realización de este proyecto son:

- El proyecto combina hardware y software. La electrónica y el software son dos materias muy diferentes pero que están fuertemente conectadas entre sí ya que se debe hacer un correcto diseño del hardware a la vez que un código eficiente para que el proyecto sea ejecutado correctamente.

- Se han empleado conocimientos adquiridos en los estudios de Grado de asignaturas como Sistemas Electrónicos Digitales para la elaboración del hardware y parte del código, de Informática, para el entendimiento de la programación a nivel lógico, o de Electrónica Analógica a la hora de realizar los circuitos electrónicos.

- Es importante hace un esfuerzo en la lectura de toda la documentación necesaria, para de esa manera comprender el correcto funcionamiento de los dispositivos utilizados y así evitar dañar los mismos.

- La realización y desarrollo de este proyecto ha proporcionado al autor nuevos conocimientos acerca de la programación en lenguaje C++, como el correcto tratamiento de las variables, además de iniciarle en la configuración y conexión inalámbrica entre dispositivos, así como con un servidor web. Además, ha conseguido

nuevos conocimientos como adentrarse en la programación en código HTML para realizar la página web, materia que no había visto anteriormente en el grado cursado.

Por el contrario, a la hora de realizar este proyecto se ha tenido que ir superando algunos problemas como:

- Conseguir que el servidor reconociese correctamente qué cliente se trataba de conectar en cada caso.

- Debido a los pocos conocimientos en lenguaje HTML, se ha tenido problemas para conseguir que la información se mostrase en la página web de una manera correcta, además de diseñar la página y conseguir una correcta funcionalidad de la misma.

8.2 LINEAS FUTURAS

Después de haber realizado el proyecto se presentan algunos aspectos con margen de mejora para el futuro. Estas posibles ampliaciones pueden ser:

- Diseñar el hardware en una PCB para reducir el tamaño del dispositivo y poder implementar el proyecto en una situación real.

- Optimizar todo lo posible la programación de los clientes y el servidor para que los programas consuman el mínimo de memoria posible, así como un consumo reducido de energía. Además de una mejora en el tratamiento de las variables usadas en el sistema.

- Realizar un diseño de la interfaz de la página web más atractiva y con más opciones de uso.

- Incorporar una batería a los clientes para poder ubicarlos en cualquier lugar sin problemas y sin necesidad de tener una fuente de energía conectada.

- Se pueden añadir nuevos sensores y actuadores al sistema de una forma muy sencilla para que su información sea visualizada en el servidor. Como, por ejemplo, en el ámbito de la salud, un lector de pulsaciones del paciente y que se mande al servidor.

- Se puede pensar en el diseño del mismo proyecto con otros dispositivos que sean más eficientes y más completos para la realización de este prototipo. Más concretamente, sería interesante la inclusión del dispositivo ESP32 ya que lleva integrado módulo WiFi y Bluetooth y es bastante eficiente para la realización de proyectos de IoT. (Constantin, 2017) (Espressif, s.f.)

9. BIBLIOGRAFÍA

Arduino, P. O. (25 de 8 de 2016). *WiFi101 library*. Obtenido de https://www.arduino.cc/en/Reference/WiFi101

Arduino, P. O. (2018). *Ethernet library*. Obtenido de https://www.arduino.cc/en/Reference/Ethernet

Arduino, P. O. (2018). *SPI library*. Obtenido de https://www.arduino.cc/en/reference/SPI

Arduino, P. o. (s.f.). *Arduino Oficial*. Obtenido de https://www.arduino.cc/en/Guide/Introduction

Arduino, P. o. (s.f.). *Genuino 101*. Obtenido de https://store.arduino.cc/genuino-101

Aruba. (s.f.). *MarketingNews.es*. Obtenido de http://www.marketingnews.es/tendencias/noticia/1104617029005/interesante-estudio-sobre-el-internet-de-las-cosas.1.html

Barrera, K. (26 de Julio de 2018). *Aprendiendo Arduino*. Obtenido de https://aprendiendoarduino.wordpress.com/2016/11/13/bluetooth-en-arduino/

Castillo, D. F. (s.f.). *Comunicaciones Inalámbricas Bluetooth*. Obtenido de http://revistas.utp.ac.pa/index.php/prisma/article/view/412/html

Constantin, B. N. (2017). *Desarrollo de un interruptor inalámbrico empleando un dispositivo WIFI ESP32.* Madrid.

"Datasheet Arduino Ethernet". Obtenido de https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf

"Descripción de Arduino Ethernet". Obtenido de https://www.arduino.cc/en/Reference/Ethernet
https://aprendiendoarduino.wordpress.com/2016/07/04/ethernet-shield/
https://store.arduino.cc/arduino-ethernet-rev3-without-poe

"Datasheet Ardino MKR1000". Obtenido de: https://www.microchip.com/wwwproducts/en/ATSAMW25

"Aprendizaje de HTML". Obtenido de: https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/HTML_basics

Espressif. (s.f.). *A different IOT power and performance*. Obtenido de https://www.espressif.com/en/products/hardware/esp32/overview

Gutierrez, M. J. (10 de 8 de 2015). *ElEspañol*. Obtenido de https://elandroidelibre.elespanol.com/2015/08/todo-sobre-zigbee-la-tecnologia-ultrabarata-para-comunicacion-inalambrica.html

“Imagen del IoT mundo”. Obtenido de:
https://hipertextual.com/2015/06/internet-of-things

“Información acerca ESP32”.

Espressif. (s.f.). *A different IoT power and performance*. Obtenido de
https://www.espressif.com/en/products/hardware/esp32/overview

Pérez Porto y Gardey. (publicado en 2008.Actualizado:2012). *Definición.De*. Obtenido de
https://definicion.de/html/

Ramírez, M. (17 de 1 de 2018). *Conoce las diferencias entre los tipos de Wifi*. Obtenido de
https://www.elgrupoinformatico.com/conoce-las-diferencias-entre-los-tipos-wifi-t41039.htm

Ruesca, P. (s.f.). *Radio Comunicaciones*. Obtenido de
http://www.radiocomunicaciones.net/radio/radio-enlace-que-es-un-radioenlace/

SAP. (s.f.). *¿qué es Internet de las Cosas(IOT)?* Obtenido de
https://www.sap.com/spain/trends/internet-of-things.html

Valencia, U. i. (s.f.). *RFID: qué es y cómo funciona*. Obtenido de
https://www.universidadviu.es/rfid-que-es/

10. ANEXOS

10.1 LENGUAJE ARDUINO

Estructuras de control

- if *(comparador si-entonces)*
- if...else *(comparador si...sino)*
- for *(bucle con contador)*
- while *(bucle por comparación booleana)*
- do... while *(bucle por comparación booleana)*
- break *(salida de bloque de código)*
- continue *(continuación en bloque de código)*
- return *(devuelve valor a programa)*

Sintaxis

- ; *(punto y coma)*
- {} *(llaves)*
- // *(comentarios en una línea)*
- /* */ *(comentarios en múltiples líneas)*
- # *(introducción de librerías)*

Operadores Aritméticos

- = *(asignación)*
- + *(suma)*
- - *(resta)*

- * *(multiplicación)*
- / *(división)*
- % *(modulo)*

Operadores Comparativos

- == *(igual a)*
- != *(distinto de)*
- < *(menor que)*
- > *(mayor que)*
- <= *(menor o igual que)*
- >= *(mayor o igual que)*

Operadores Booleanos

- && *(AND)*
- || *(OR)*
- ! *(negación)*

Operadores compuestos

- ++ *(incremento)*
- -- *(decremento)*
- += *(suma compuesta)*
- -= *(resta compuesta)*
- *= *(multiplicación compuesta)*
- /= *(división compuesta)*

Constantes

- HIGH | LOW *(Nivel alto / Nivel bajo)*
- INPUT | OUTPUT *(Entrada/ Salida)*
- true | false *(Verdadero/ Falso)*
- Constantes Numéricas

Tipos de Datos

- boolean *(1,0,true,false)*
- char *(carácter ej. 'a' o de -128 a 127)*
- byte *(de 0 a 255)*
- int *(entero de -32768 a 32767)*
- unsigned int *(entero sin signo de 0 a 65535)*
- long *(entero 32b, de -2147483648 a 2147483647)*
- unsigned long *(entero 32b sin signo de 0 a 4294967295)*
- float *(en coma flotante de -3.4028235E +38 a 3.4028235E +38)*
- double *(en coma flotante de 32b de -3.4028235E +38 a 3.4028235E +38)*
- String *(cadena de caracteres)*
- array *(cadena)*
- void *(vacío)*

E/S Digitales

- PinMode *(Pone un pin como entrada/salida)*
- DigitalWrite *(escribe en un pin de salida un valor)*
- DigitalRead *(Lee el valor por un pin de entrada)*

E/S Analógicas

- analogRead (Led entrada analógica por un pin)
- analogWrite (*Escribe por un pin de salida analógica)*

Tiempo

- millis() (*devuelve el número de milisegundos desde que se ha reseteado*)
- micros() – (*devuelve el número de milisegundos desde que se ha reseteado*)
- delay() *(Pausa el programa los milisegundos introducidos en la función)*

Matemáticas

- min(x,y) *(mínimo)*
- max(x,y) *(máximo)*
- abs(x) *(valor absoluto)*
- constrain(x, valMin, valMax) *(limita)*
- pow(base, exponente) *(eleva a un número)*
- sq(x) *(eleva al cuadrado)*
- sqrt(x) *(raíz cuadrada)*

Comunicación Serial

- Serial.begin(velocidad) – *(abre puerto serie y establece velocidad)*
- Serial.end() – (*desactiva puerto serie)*
- Serial.available() – *(rebela si hay datos en el Puerto)*
- Serial.read() – *(lee un único carácter del buffer)*

- Serial.flush() – *(vacía el buffer)*
- Serial.print(datos) – *(imprime los datos en el puerto serie)*
- Serial.println(datos) – *(imprime los datos en el puerto serie con retorno de carro)*
- Serial.write(bytes) – (*escribe caracteres en el puerto serie)*

Otras Funciones de interés incluidas en el programa

- indexOf() *(Localiza un carácter o una cadena dentro de otra cadena)*
- Ethernet.begin() *(Inicializa la librería Ethernet.h)*
- WiFi.begin() *(Inicializa la librería WiFi101.h)*
- server.available() *(Obtiene un cliente que está conectado al servidor y tiene datos disponibles para su lectura)*
- client.available() *(Devuelve el número de bytes disponibles para la lectura. Es decir, la cantidad de datos que se han escrito por el cliente para el servidor al que está conectado).*
- client.println() *(Imprime los datos, seguidos de un retorno de carro y salto de línea, al servidor de un cliente que está conectado)*
- client.stop() *(Desconecta el cliente del servidor)*
- Ethernet.localIP() *(Obtiene la dirección IP del Arduino Ethernet)*
- EthernetClient() *(Crea un cliente que puede conectarse a una dirección IP y puerto de Internet especificados mediante Ethernet)*
- WiFiClient() *(Crea un cliente que puede conectarse a una dirección IP y puerto de Internet especificados mediante WiFi)*
- Client.flush() *(Se limpia el buffer del cliente)*

10.2 CÓDIGO REALIZADO

10.2.1 CLIENTE 1

```
#include <SPI.h>
#include <WiFi101.h>
#include "arduino_secrets.h"
char ssid[] = "MOVISTAR_7C58";        // your network SSID (name)
char pass[] = "ECB2996D867E70DBBCC6"; // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0;  // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
// Informacion de nuestro servidor
const char host[] = "192.168.1.37"; //Direccion IP de nuestro servidor
const int httpPort = 80; //Puerto de enlace
int PIN =3;
int ROJO= 5;
int VERDE=4;
int AMARILLO=6;
int pulsador = 0;
String macID= "D0A3"; //String donde se almacenará nuestra macID
String line;

void setup() {
   //Inicializo los pines de salida y entrada
  pinMode(3, INPUT); //pulsador como entrada
  digitalWrite(3, LOW); //se inicia a nivel bajo
  pinMode(5, OUTPUT); // led rojo
  digitalWrite(5, LOW);
   pinMode(4, OUTPUT);//led verde
  digitalWrite(4, LOW);
  pinMode(6, OUTPUT);//led amarillo
```

```
 digitalWrite(6, LOW);
  Serial.print(macID);
 delay(10);
 Serial.begin(9600);      // initialize serial communication
 printWiFiStatus();    //conectar a WiFi
}

void loop() {
 delay(1);
 pulsador= digitalRead(3);
 Serial.print("el pulsador tiene el valor: ");
 Serial.println(pulsador);
 if (pulsador==1){
 digitalWrite(ROJO, LOW);

 digitalWrite(AMARILLO, HIGH);
 }
// Inicia la conexion con el servidor
 Serial.print("connecting to ");
 Serial.println(host);

 // Se intenta la conexion con el servidor
 WiFiClient client;
 if (!client.connect(host, httpPort)) {
  Serial.println("connection failed");
  return;
 }
 String url = macID;
 url += "pulsador=";
 url += pulsador;
```

```
 Serial.print("Requesting URL: ");
 Serial.println(url);

  // Se envia la solicitud al servidor
 client.print(String("GET /") + url + "/ HTTP/1.1\r\n");

 client.flush(); //limpiar el buffer
   unsigned long timeout = millis();
 while (client.available() == 0) {
  if (millis() - timeout > 5000) {
    Serial.println(">>> Client Timeout !");
    client.stop();
    return;
  }
 }
    while(client.available()){
    line = client.readStringUntil('\r');
  Serial.println(line);
  if (line == "recibido1"){
 digitalWrite(VERDE, HIGH);
 digitalWrite(ROJO, LOW);
 digitalWrite(AMARILLO, LOW);
 Serial.println("enciendo el led");
   }

}

   // Se lee todo lo recibido desde el servidor y se imprime en el puerto serial
 Serial.println();
 Serial.println("closing connection");
}
```

```
void printWiFiStatus() {
    // check for the presence of the shield:
if (WiFi.status() == WL_NO_SHIELD) {
  Serial.println("WiFi shield not present");
  while (true);       // don't continue
}
// attempt to connect to WiFi network:
while ( status != WL_CONNECTED) {
  Serial.print("Attempting to connect to Network named: ");
  Serial.println(ssid);                   // print the network name (SSID);

  // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
  status = WiFi.begin(ssid, pass);
  // wait 10 seconds for connection:
  delay(10000);
}
// print the SSID of the network you're attached to:
Serial.print("SSID: ");
Serial.println(WiFi.SSID());

// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);

// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");
```

```
digitalWrite(ROJO, HIGH);
}
```

10.2.2. CÓDIGO SERVIDOR

```
#include <Ethernet.h>
#include <SPI.h>
byte mac[] = { 0x90, 0xA2, 0xDA, 0x11, 0x41, 0x26 };
EthernetServer server(80);
int Pulsador1;
int Pulsador2;
int a=0;
int z=0;

void setup() {
  config_salidas ();
    // start the serial library:
  Serial.begin(9600);
  // start the Ethernet connection:
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    // no point in carrying on, so do nothing forevermore:
    for(;;)
      ;
  }
  // print your local IP address:
  Serial.println(Ethernet.localIP());
  server.begin();
}
void loop() {
  delay(1);
String s=" ";

  // Si el cliente está conectado
```

```
 EthernetClient client = server.available();
 if (!client) {
Serial.println("no hay cliente");
  return;
 }

 // Se realiza la lectura de la solicitud
 String req = client.readStringUntil('\r');
 Serial.println(req);
 //String req1=req;

 client.flush();

 int val = -1;
 if (req.indexOf("/D0A3") != -1){  //Dispositivo 1;

  String value1;
  val = 0;
  int Start1 = req.indexOf("=");
  int Finish1 = req.indexOf('/', Start1 + 1);
  Serial.print("start1: ");
  Serial.println(Start1);
  Serial.print("Finish1: ");
  Serial.println(Finish1);
  value1 = "";

  for (int i = Start1 + 1; i < Finish1; i++)
{
  value1 = value1 + req.charAt(i);
}
```

```
  Pulsador1=value1.toInt();


   Serial.print("el pulsador1 tiene el valor: ");
   Serial.println(Pulsador1);
 if (Pulsador1==1){
  digitalWrite(6, LOW);
  digitalWrite(9, LOW);
 }
    else if ((Pulsador2==0) && (Pulsador1==0) && (a==1)){
digitalWrite(8, HIGH);
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(9, HIGH);
digitalWrite(5, HIGH);
digitalWrite(6, HIGH);
digitalWrite(7, HIGH);
a=0;
z=1;
   }
}
  if (req.indexOf("/ECF0") != -1){  //Dispositivo 2;
    String value2;
   val = 1;
   int Start2 = req.indexOf("=");
   int Finish2 = req.indexOf('/', Start2 + 1);
   Serial.print("start2: ");
   Serial.println(Start2);
   Serial.print("Finish2: ");
   Serial.println(Finish2);
     value2 = "";
```

```
  for (int b = Start2 + 1; b < Finish2; b++)
 {
  value2 = value2 + req.charAt(b);
 }
 Serial.println(value2);

  Pulsador2=value2.toInt();

  if (Pulsador2==1){
 digitalWrite(7, LOW);
 digitalWrite(2, LOW);
 digitalWrite(5, LOW);
 digitalWrite(6, LOW);
 digitalWrite(8, LOW);
  }
  else if ((Pulsador2==0)&&(Pulsador1==0) && (a==2)){//hasta que no se da la señal desde el
serv que se ha recibido no se apaga el display
digitalWrite(8, HIGH);
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(9, HIGH);
digitalWrite(5, HIGH);
digitalWrite(6, HIGH);
digitalWrite(7, HIGH);
a=0;
z=2;
  }
 }

  else if (req.indexOf("/menu") != -1){
 val = -2; // Se imprimira el menu de la pagina web
```

```
 Serial.println("se ha dado al menu");
 }
  else if (req.indexOf("/?var=ACTIVAR1") != -1){
 val = -3; // Se imprimira aviso paciente 1
 a=1;
  }
 else if (req.indexOf("/?var=ACTIVAR2") != -1){
 val = -4; // Se imprimira aviso paciente2
 a=2;
 }

// Se envia la respuesta a nuestro cliente

if ((val>=0)&&(z==1)){

      s="recibido1";
      Serial.println("s1\r\n");
      client.println("recibido1");
      delay(1);
      z=0;
  }
  if ((val>=0)&&(z==2)){

      s="recibido2";
      Serial.println("s2\r\n");
      client.println("recibido2");
      delay(1);
      z=0;
         }

  else if (val < -1){
```

```
Serial.println("Iniciamos un header comun");

// Si nuestro clIente es otro dispositivo


// Se prepara la respuesta. Iniciamos con un header comun:

client.println("HTTP/1.1 200 OK\r\n");

 client.println("<!DOCTYPE HTML>\r\n<html>\r\n");


if (val == -2) {


  //Serial.println("menu para elegir dispositivo");

   // Se imprimen en una pagina web el menu para elegir que dispositivo se desea realizar la lectura

   client.println("<head><title>Arduino Ethernet WEB SERVER .</title></head>");

   client.println("<body><center><br><br><br><h1><u> P&aacute;gina Web que refleja los dispositivos activados </u></h1>");

    client.println("<form action=\"192.168.1.37/\" method=\"GET\">");

   client.println("<meta http-equiv=\"refresh\" content=\"1\"></head>"); //Funcion que realiza el refrescamiento cada segundo

   client.println("<center><h2> Se puede mandar la se&ntilde;al de socorro al activarse los pulsadores</h2>");


   client.println("Pulsador1 = "  + String(Pulsador1) + "<br>");

      client.println("<input type=\"submit\" name=\"var\"
     value=\"ACTIVAR1\"> Dispositivo 1.<br>");


   client.println("<br>");

   client.println("Pulsador2 = "  + String(Pulsador2) + "<br>");

      client.println("<input type=\"submit\" name=\"var\"
     value=\"ACTIVAR2\"> Dispositivo 2.<br>");


 }

      if (val == -3) {
```

```
 // cliente1
 client.println("<head><title>Mediciones de dispositivo 1.</title>");
 client.println("<meta http-equiv=\"refresh\" content=\"1\"></head>");
 client.println("<center><h2> SE HA MANDADO EL AVISO AL PACIENTE 1</h2>");
 client.println("<p><em> Volver al menu principal.</em></p></body></center>");

}
if (val == -4) {

 // cliente 2
 client.println("<head><title>Mediciones de dispositivo 2.</title>");
 client.println("<meta http-equiv=\"refresh\" content=\"1\"></head>");
 client.println("<center><h2> SE HA MANDADO EL AVISO AL PACIENTE 2</h2>");
 client.println("<p><em> Volver al menu principal.</em></p></body></center>");
}

client.println("</html>\n");
}
else
{
 client.println("Invalid Request.<br>");
}



Serial.println(val);
Serial.print("a vale ");
Serial.println(a);
delay(10);

 Serial.print("Req vale: ");
```

```
  Serial.println(req);
  client.stop();
  Serial.println("Cliente desconectado");
  Serial.println("");
  Serial.println("");
}
void config_salidas (){
  //Se configuran los pines fisicos de entradas y salidas
  pinMode(4,OUTPUT);                    // CONFIGURA EL PIN 4 COMO SALIDA DIGITAL (sd)
 digitalWrite(4,HIGH);                  // COLOCA EL PIN 4 DE LA ARDUINO EN UN NIVEL
LÓGICO ALTO

// es display de anodo comun: se activa a nivel bajo(hay que iniciarlos a nivel alto)
pinMode(6, OUTPUT);    //b
pinMode(2, OUTPUT); //g
pinMode(3, OUTPUT); //f
pinMode(9, OUTPUT);    //c
pinMode(8, OUTPUT);//d
pinMode(5, OUTPUT);//a
pinMode(7, OUTPUT);     //e
digitalWrite(8, HIGH);
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(9, HIGH);
digitalWrite(5, HIGH);
digitalWrite(6, HIGH);
digitalWrite(7, HIGH);
}
```