ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE
TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Trabajo Fin de Grado

Desarrollo de un servidor web asíncrono con Arduino

AUTOR: JOSE CARLOS GONZÁLEZ VIDAL
DIRECTOR: DR. FERNANDO LOSILLA LÓPEZ
OCTUBRE/ 2015

Autor	Jose Carlos González Vidal
E-mail del Autor	**txecar@hotmail.com**
Director	Fernando Losilla López
E-mail del Director	fernando.losilla@upct.es
Codirector(es)	
Título del TFG	Desarrollo de un servidor web asíncrono con Arduino
Descriptores	

Resumen

El trabajo que se presenta en la memoria de este proyecto fin de carrera se adentra en la implementación de un servidor web asíncrono sobre Arduino el cual es capaz de controlar sensores y actuadores conectados a la plataforma Arduino.

A lo largo de esta memoria se describirán los distintos protocolos y estándares que han sido tenidos en cuenta a la hora de su realización. Se haciendo una descripción del hardware Arduino MEGA 2560 que soportara el servidor y los actuadores, también cabe destacar el Ethernet Shield 2 que permitirá las comunicaciones. Para culminar se describirá el funcionamiento de la aplicación servidor.

Titulación	Grado en Ingeniería Telemática
Intensificación	
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Octubre 2015

Agradecimientos

Agradecer a mi director de proyecto Fernando por permitirme volver a realizar un proyecto con él, sé que hay veces que parecía que no me enteraba de lo que me explicaba y en ocasiones era necesario volver a verlo más de una vez, gracias por tu paciencia.

Me gustaría agradecer a mis compañeros y amigos por apoyarme en las decisiones que he tomado a la hora de realizar la pasarela y que si no hubieran estado conmigo es posible que no la hubiera acabado.

Por último me gustaría agradecer a mi familia por todo el apoyo que me siguen dando y por todo el esfuerzo que ponen en que mejore en todo lo que hago.

Sinceramente gracias a todos.

INDICE

1 - Introducción

La utilización de Internet en nuestro día a día ha aumentado de manera exponencial desde hace unas décadas. No hay día en que uno no se pueda levantar de la cama y no mirar el correo electrónico para comprobar si hay algún mensaje nuevo. Esto nos lleva a pensar que si podemos comunicarnos con cualquier persona que se encuentre en cualquier lugar del mundo con una conexión a Internet porque no podernos comunicarnos con nuestros aparatos electrónicos.

El hecho de poder decidir si las condiciones de un entorno son correctas o no para, por ejemplo, la instauración de nuevas plantaciones de cultivo o simplemente bajar una persiana por exceso de luz o porque ya no aporta luz suficiente, puede ser decidido gracias a pequeños sensores que tomen medidas de la cantidad de luz que recibe la zona conectados por ejemplo a un dispositivo como Arduino.

Con el uso de Arduino, una plataforma de hardware y software de código abierto (open-source), que lleva en su placa un microcontrolador y ofrece además su propio entorno de desarrollo, se pretende proporcionar Internet a ciertos aparatos o dispositivos. El hecho de tener código abierto da grandes posibilidades a los desarrolladores para crear proyectos que corran sobre ellos, ya que no es necesaria una licencia para su creación. El código abierto junto con el bajo coste de la plataforma Arduino y de los dispositivos que pueden ser conectados a ella, ha llevado a una gran proliferación de su uso. Estos usos pueden llevarse a la docencia para la creación de prototipos que controlen robots, coches teledirigidos, etc. ya que no es necesario que estos estén conectados a un ordenador para su funcionamiento.

Por estas razones se plantea la realización de este proyecto, en el que se pretende desarrollar un servidor web asíncrono sobre Arduino con el que se puedan controlar en tiempo real sensores y actuadores conectados a una plataforma Arduino, gracias a que veremos si se producen cualquier modificación en los valores aportados por los sensores o actuadores.

Para el desarrollo de este proyecto se plantean una serie de tareas y retos que deben abordarse:

- Aprendizaje de Ajax para su posterior uso en el servidor web.
- Aprendizaje de las librerías para el control de Arduinos a través de Ethernet.
- Procesado de peticiones HTTP. Dado que HTTP es el protocolo fundamental en la Web se debe desarrollar un sistema capaz de procesar correctamente peticiones HTTP básicas en dispositivos con capacidades limitadas. En este proyecto se procesan los métodos GET y POST que harán las llamadas al servidor web.
- Creación del servidor que devolverá de manera asíncrona los valores de sensores y actuadores a traces de una web.

2 - Estándares y Protocolos

2.1 Introducción

Como primer paso hacia la realización de este proyecto final de carrera se ha de realizar un estudio de los estándares y protocolos que se deberán implementar para el correcto funcionamiento del mismo.

2.2 HTTP

HTTP es un protocolo de transferencia de hipertexto (HyperText Transfer Protocol) que trabaja a nivel de aplicación utilizado para la trasferencia de información. HTTP define la sintaxis y semántica que es utilizada para la comunicación entre los diversos elementos de la arquitectura del software. Este protocolo es usado por arquitecturas cliente-servidor con un esquema petición-respuesta.

El protocolo sigue una especie de esquema que podría ser el siguiente. Un cliente realiza una petición que estará formada por un método, una URI, y una versión del protocolo en la primera línea, a continuación se mandaran las cabeceras de la petición y por último un posible dato a introducir. El servidor contesta con una línea de estado que incluye la línea del protocolo y un código de éxito o error, en las siguientes líneas enviara las cabeceras de la petición y por último y si fuera necesario algún tipo de datos.

HTTP es un protocolo sin estado, lo que significa que no guarda ninguna información de sus conexiones anteriores. Para dar una solución a este problema se crearon las cookies, las cuales son información que el servidor guarda en los clientes. Las cookies abren un abanico de posibilidades como la creación de sesiones, la compra por Internet.

Petición

HTTP define 8 métodos: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS y CONNECT.

GET

Pide una representación del recurso especificado. Por seguridad no debería ser usado por aplicaciones que causen efectos ya que transmite información a través de la URI agregando parámetros a la URL.

HEAD

El método HEAD es igual que el método GET, salvo que el servidor sólo tiene que devolver las cabeceras. Estas cabeceras son las mismas que si se utilizara el método GET.

POST

El método POST es utilizado para realizar peticiones en las que el servidor destino acepta el contenido de la petición como un nuevo subordinado del recurso pedido. Otro de sus usos es el de cambiar el estado del servidor.

PUT

El método PUT sube, carga o realiza un upload de un recurso, es la manera más eficiente de subir archivos a un servidor.

DELETE

El método DELETE es utilizado para que el servidor borre un recurso indicado por la URI de la petición.

TRACE

El método TRACE es utilizado para comprobar la existencia del receptor del mensaje y usar la información para realizar un diagnostico.

OPTIONS

El método OPTIONS devuelve los métodos HTTP que el servidor soporta para un URL específico. Esto puede ser utilizado para comprobar la funcionalidad de un servidor web mediante petición.

CONNECT

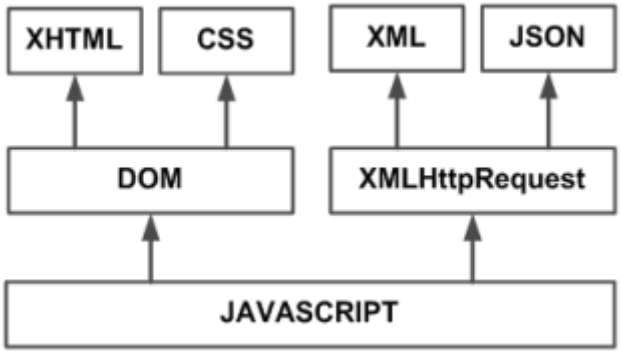
El método CONNECT es utilizado para saber si un proxy nos da acceso a un host bajo condiciones especiales.

2.3 AJAX

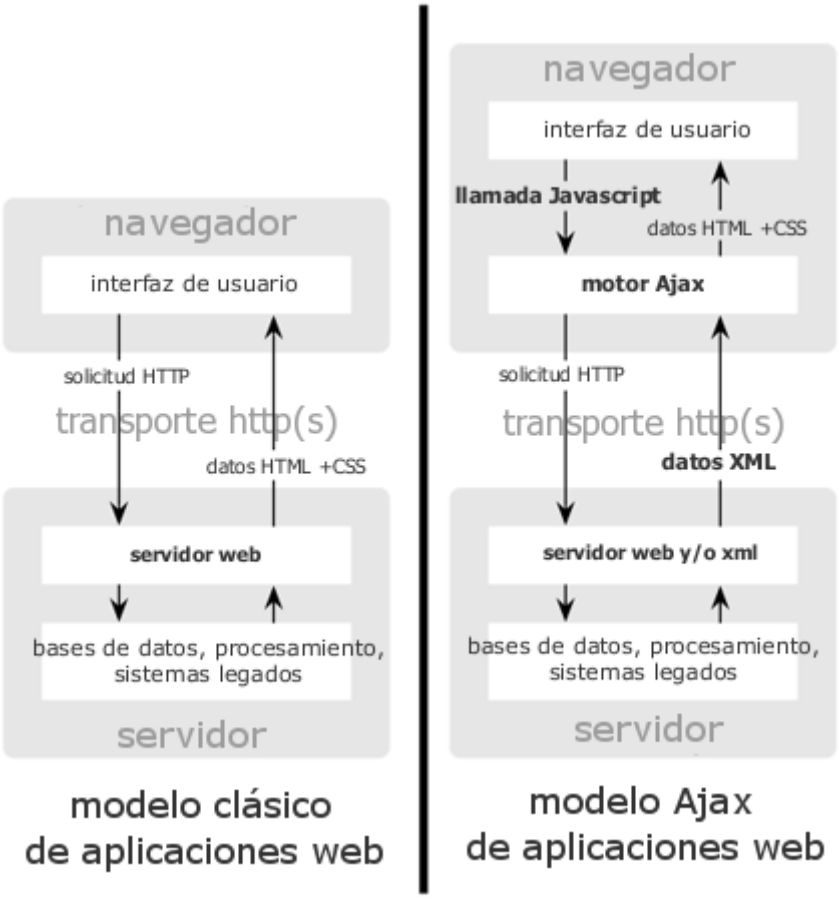
AJAX es el acrónimo de Asynchronous JavaScript + XML que puede ser traducido como "JavaScript asíncrono + XML". AJAX es el término que describe la posibilidad de hacer peticiones al servidor sin tener que volver a cargar la página y la posibilidad de trabajar y analizar documentos XML.

AJAX no es una tecnología por sí mismo, sino que se trata de la unión de varias tecnologías independientes, tales como:

- XHTML y CSS, que crearan la presentación.
- DOM, que se encargara de la manipulación e interpretación dinámica de la presentación.
- XML, XSLT y JSON, que serán encargadas de la manipulación de información.
- XMLHttpRequest, encargada del intercambio asíncrono de información.
- JavaScript, será el nexo de unión del resto de tecnologías.



Al hacer uso de AJAX no es necesario recargar toda la página cuando se picha en un enlace o en un botón, ya que con AJAX es posible hacer una conexión al servidor desde la página web usando un programa JavaScript mediante XMLHttpRequest.



Ventajas y Desventajas de AJAX

-Ventajas:

- No es necesario recargar toda la página, por lo que el tiempo de respuesta es reducido
- Las comunicaciones se realizan en segundo plano por lo que no hay interrupciones.

-Desventajas:

- El usar el botón de avance o volver atrás pueden dejar de estar disponibles.
- Desarrollo de páginas web más complejo.
- Problemas y restricciones de seguridad relacionados con el uso de AJAX.

2.4 HTML

HTML o HiperText Markup Languaje (Lenguaje de Marcación de Hipertexto) es un lenguaje de programación utilizado fundamentalmente para el desarrollo de páginas web. Más concretamente establece la estructura y contenido de la web, del texto, objetos e imágenes. Es un estándar de W3C (World Wide Web), organización dedicada a la estandarización de la mayor parte de las tecnologías web.

HTML hace uso de “etiquetas” para describir el funcionamiento y la apariencia del texto que es etiquetado. Las etiquetas van especificadas entre corchetes o paréntesis angulares: <>. También cabe destacar el uso de scripts, que son encargados de aportar instrucciones específicas a los navegadores los cuales procesaran el lenguaje. Los script más utilizados son JavaScript y PHP.

Códigos HTML básicos:

<html>: define el inicio del documento e indica al navegador que las siguientes líneas pertenecen a un documento HTML.

<script>: incrusta un script en la web.

<head>: define la cabecera del documento.

<body>: define el contenido principal del documento. Es aquí donde se encuentra lo que será mostrado mediante el uso de un navegador.

2.5 CSS

CSS son las siglas de Cascading Style Sheets u Hojas de Estilo en Cascada. Es un lenguaje desarrollado para organizar la presentación y aspecto de una página web y separarlo de la estructura de esta. Con ello se podrá separar el contenido de los documentos escritos en HTML, XML y XHTM de la presentación de este la cual incluiría elementos tales como colores, fondos, bordes, tipos de letra, etc. modificando así la apariencia de la página web.

CSS se basa en unas reglas las cuales consisten en un selector y una declaración, la cual va entre corchetes y contiene las propiedades o atributos, y un valor separados por dos puntos.

Ejemplo:

```
h1{color:red;}
```

Donde “h1” es el selector y éste se le asignará a los elementos HTML que se deseen que sean afectados por esta propiedad, “color” es la propiedad o atributo y “red” es el valor.

Podemos diferenciar el estilo según la manera de proporcionar la información de CSS, ya sea adjuntando por separado o incorporado en el documento HTML. Cabe destacar tres formas de dar estilo a un documento web: hoja de estilo externa, hoja de estilo interna, estilo en línea.

Ventajas de CSS

- Separación del aspecto de la estructura
- Ahorro de ancho de banda debido a que la hoja es estilo se guarda en caché después de la primera solicitud.

2.6 XML

XML, abreviatura de eXtensible Markup Language (Lenguaje de Etiquetado Extensible), es un lenguaje estándar que permite escribir datos estructurados en un fichero de texto, desarrollado por el World Wide Web Consortium. Al ser "Extensible" no posee una limitación de etiquetas, por lo que se pueden crear tantas como sean necesarias.

XML es una adaptación de SGML, el cual es el lenguaje de marcas estándar para describir el formato y el contenido de los documentos. XML surge debido a que HTML era un lenguaje poco efectivo para negocios virtuales.

Ventajas de la utilización de XML:

- Es extensible.
- Su analizador es estándar(no requiere de cambios para cada versión de metalenguaje).
- Facilita el análisis y procesamiento de documentos XML creados por terceros.

2.7 JavaScript

JavaScrip es un lenguaje de programación interpretado, por lo que no es necesario compilarlo para ejecutarlo. Surgió para ampliar las posibilidades del HTML y es el encargado de realizar tareas y operaciones en el marco de la aplicación cliente. El hecho de que surgiera relativamente pronto es la causa más probable de que se haya convertido en un estándar, el cual es soportado por todos los navegadores actuales.

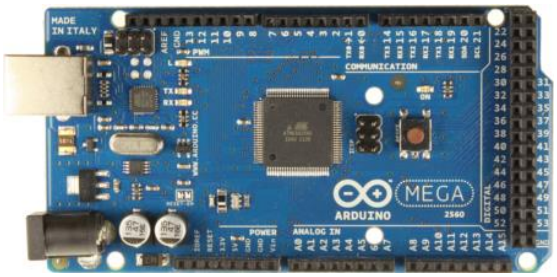
Contrariamente a lo que se pueda creer JavaScript no es interpretado en Java. Es un lenguaje basado en objetos cuya sintaxis básica es semejante a la de Java y C++ para reducir el número de conceptos necesarios para aprender el idioma. Como ejemplo, los bucles while y switch funcionan del mismo modo que en estos lenguajes.

3 - Entorno de Trabajo

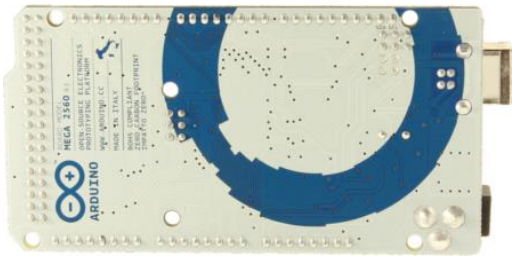
3.1 Introducción

En este capítulo se pretenden describir las principales herramientas hadware y software sobre las que se ha llevado a cabo este proyecto fin de carrera, de una manera detallada.

3.2 Arduino Mega 2560

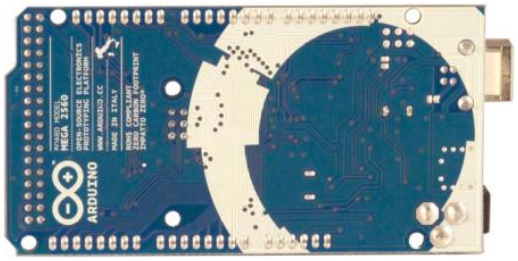


Arduino Mega 2560 R3 delantero



Arduino Mega2560 R3

Arduino Mega 2560 Frontal



Arduino Mega 2560

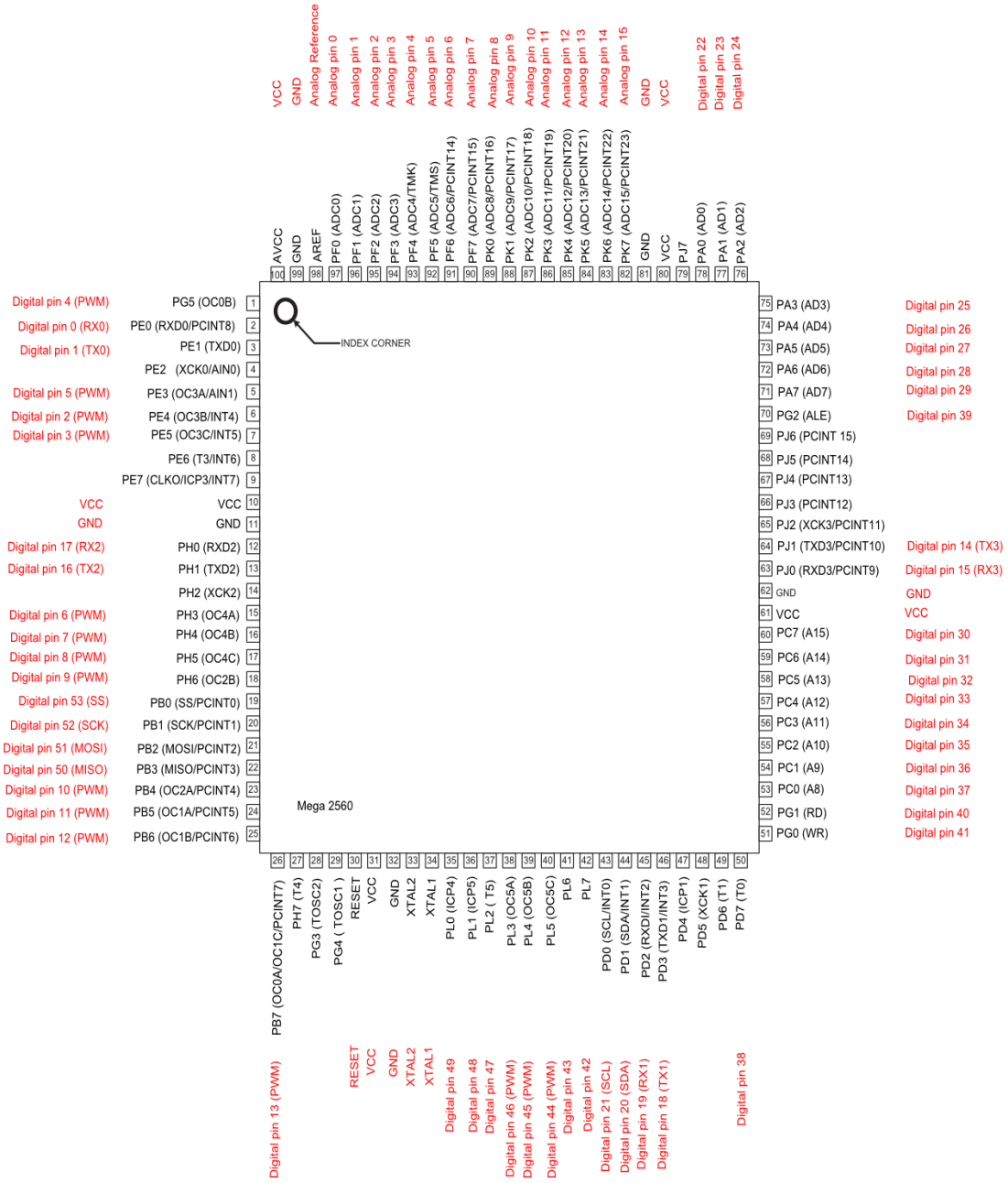
3.2.1 Vision General

El Arduino Mega 2560 es una placa mmicrocontrolador basada en el microprocesador Atmega2560. Tiene 54 entradas/salidas digitales (de las cuales 15 pueden utilizarse para salidas PWM), 16 entradas analógicas, 4 UARTs (puertos serie por hardware), un oscilador de 16MHz, una conexión USB, entrada de corriente, conector ICSP y botón de reset. Contiene todo lo necesario para hacer funcionar el microcontrolador, simplemente conectarlo a un ordenador con un cable USB, o alimentarlo con un adaptador de corriente AC a DC para empezar. Mega es compatible con la mayoría de los shield diseñados para Arduino Duemilanove o Diecimila.

Mega 2560 es una actualización de la placa Arduino Mega , al que sustituye.

Mega2560 difiere de todas las placas anteriores ya que no utiliza el chip controlador de USB a serial FTDI. En su lugar, ofrece el ATmega16U2 programado como convertidor USB a serie. Esto implica que se trabaje con /dev/ACM.

3.2.2 Mapeado de Pin



3.2.3 Resumen

Microcontrolador	ATmega2560
Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Pines E/S digitales	54 (15 proporcionan salida PWM)
Pines de entrada analógica	16
Intensidad por pin	40 mA
Intensidad en pin 3.3V	50 mA
Memoria Flash	256 KB de las cuales 8 KB las usa el gestor de arranque(bootloader)
SRAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz

3.2.4 Alimentación

El Arduino Mega puede ser alimentado por la conexión USB o por una fuente de alimentación externa. El origen de la alimentación es seleccionado automáticamente.

Las fuentes de alimentación externas (no-USB) pueden ser un transformador o una batería. El transformador se puede conectar usando un conector macho de 2.1mm con centro positivo en el conector hembra de la placa. Los cables de la batería pueden conectarse a los pines GND y Vin en los conectores de alimentación (POWER).

La placa puede trabajar con una alimentación externa de entre 6 a 20 voltios. Si el voltaje suministrado es inferior a 7V el pin de 5V puede proporcionar menos de 5 Voltios y la placa puede volverse inestable, si se usan mas de 12V los reguladores de voltaje se pueden sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 voltios.

Los pines de alimentación son los siguientes:

- **VIN.** La entrada de voltaje a la placa Arduino cuando se está usando una fuente externa de alimentación (en contraposición a los 5 voltios de la conexión USB). Se puede proporcionar voltaje a través de este pin, o, si se está alimentado a través de la conexión de 2.1mm, acceder a ella a través de este pin.

- **5V.** La fuente de voltaje estabilizado usado para alimentar el microcontrolador y otros componentes de la placa. Esta puede provenir de VIN a través de un regulador

integrado en la placa, o proporcionada directamente por el USB u otra fuente estabilizada de 5V.

- **3V3.** Una fuente de voltaje a 3.3 voltios generada en el chip FTDI integrado en la placa. La corriente máxima soportada 50mA.

- **GND.** Pines de toma de tierra.

- **IOREF.** Este pin proporciona la referencia de tensión con la que opera el microcontrolador. Un shield configurado puede leer el voltaje pin IOREF y seleccionar la fuente de alimentación adecuada o habilitar traductores tensión en las salidas para trabajar con los 5V o 3.3V.

3.2.5 Memoria

El ATmega2560 tiene 256KB de memoria flash para almacenar código (8KB son usados para el arranque del sistema (bootloader)).El ATmega2560 tiene 8 KB de memoria SRAM y 4KB de EEPROM.

3.2.6 Entradas y Salidas

Cada uno de los 54 pines digitales en el Mega pueden utilizarse como entradas o salidas usando las funciones pinMode(), digitalWrite(), y digitalRead() . Las E/S operan a 5 voltios. Cada pin puede proporcionar o recibir una intensidad máxima de 40mA y tiene una resistencia interna (desconectada por defecto) de 20-50kOhms. Además, algunos pines tienen funciones especializadas:

- **Serie: 0 (RX) y 1 (TX), Serie 1: 19 (RX) y 18 (TX); Serie 2: 17 (RX) y 16 (TX); Serie 3: 15 (RX) y 14 (TX).** Usado para recibir (RX) transmitir (TX) datos a través de puerto serie TTL. Los pines Serie: 0 (RX) y 1 (TX) están conectados a los pines correspondientes del chip ATmega16U2.

- **Interrupciones Externas: 2 (interrupción 0), 3 (interrupción 1), 18 (interrupción 5), 19 (interrupción 4), 20 (interrupción 3), y 21 (interrupción 2).** Estos pines se pueden configurar para lanzar una interrupción en un valor LOW (0V), en flancos de subida o bajada (cambio de LOW a HIGH (5V) o viceversa), o en cambios de valor.

- **PWM: de 2 a 13 y 44 a 46.** Proporciona una salida PWM (Pulse Wave Modulation, modulación de onda por pulsos) de 8 bits de resolución (valores de 0 a 255) a través de la función analogWrite().

- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** Estos pines soportan comunicación SPI utilizando la biblioteca de SPI . Los pines SPI también se desglosan en la cabecera ICSP, que es físicamente compatible con el Uno, Duemilanove y Diecimila.

- **LED: 13.** Hay un LED integrado en la placa conectado al pin digital 13, cuando este pin tiene un valor HIGH(5V) el LED se enciende y cuando este tiene un valor LOW(0V) este se apaga.

- **TWI: 20 (SDA) y 21 (SCL)** Apoyar la comunicación TWI utilizando la librería Wire.

El Mega2560 tiene 16 entradas analógicas, cada una de las cuales proporcionan una resolución de 10 bits (1.024 valores diferentes). Por defecto se miden desde el tierra a 5 voltios, aunque es posible cambiar la cota superior de su rango utilizando el pin AREF y función analogReference ().

Hay unos otros pines en la placa:

- **AREF.** Voltaje de referencia para las entradas analógicas. Usado por analogReference().

- **Reset.** Suministrar un valor LOW (0V) para reiniciar el microcontrolador. Típicamente usado para añadir un botón de reset a los shields que no dejan acceso a este botón en la placa.

3.2.7 Comunicación

El Arduino Mega2560 tiene una serie de facilidades para la comunicación con un ordenador, otro Arduino, u otros microcontroladores. El ATmega2560 proporciona cuatro UART hardware para TTL (5V) de comunicación serie. Un ATmega16U2 canaliza a uno de ellos sobre el USB y proporciona un puerto com virtual para software al equipo (máquinas de Windows tendrá un archivo .inf, pero las máquinas OSX y Linux reconocerán la placa automáticamente). El software de Arduino incluye un monitor serie que permite enviar datos desde y hacia la placa. Los LEDs RX y TX de la placa parpadearán cuando se están transmitiendo datos a través de ATmega8U2/ATmega16U2 chip y la conexión USB al ordenador (pero no para la comunicación serial en los pines 0 y 1).

La biblioteca SoftwareSerial permite la comunicación en serie en cualquiera de los pines digitales del Mega2560.

El ATmega2560 también soporta TWI y la comunicación SPI.

3.2.8 Programación

El ATmega2560 en el Arduino Mega viene precargado con un gestor de arranque (bootloader) que permite cargar nuevo código sin necesidad de un programador por hardware externo. Se comunica utilizando el protocolo STK500 original.

También se puede saltar el gestor de arranque y programar directamente el microcontrolador a través del puerto ISCP (In Circuit Serial Programming).

3.2.9 Reinicio Automático por Software

En vez de necesitar reiniciar presionando físicamente el botón de reset antes de cargar, el Arduino Mega2560 está diseñado de manera que es posible reiniciar por software desde el ordenador al que esté conectado. Una de las líneas de control de flujo (DTR) del ATmega8U2 está conectada a la línea de reinicio del ATmega2560 a través de un condensador de 100 nanofaradios. Cuando la línea se pone a LOW (0V), la línea de reinicio también se pone a LOW el tiempo suficiente para reiniciar el chip. El software de Arduino utiliza esta característica para permitir cargar los sketches con solo apretar un botón del entorno. Dado que el gestor de arranque tiene un lapso de tiempo para ello, la activación del DTR y la carga del sketch se coordinan perfectamente.

Esta configuración tiene otras implicaciones. Cuando el Mega2560 se conecta a un ordenador con Mac OS X o Linux, este reinicia la placa cada vez que se realiza una conexión desde el software (vía USB). El medio segundo aproximadamente posterior, el gestor de arranque se ejecutará. A pesar de estar programado para ignorar datos mal formateados (p.ej. cualquier cosa que la carga de un programa nuevo) intercepta los primeros bytes que se envían a la placa justo después de que se abra la conexión. Si un sketch que se está ejecutando en la placa recibe algún tipo de configuración inicial u otro tipo de información al inicio del programa, es necesario asegurarse de que el software con el cual se comunica espera un segundo después de abrir la conexión antes de enviar los datos.

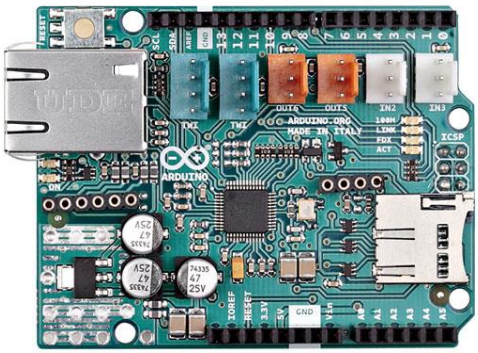
El Mega2560 contiene una pista que puede ser cortada para deshabilitar el auto-reset. Las terminaciones a cada lado pueden ser soldadas entre ellas para rehabilitarlo. Están etiquetadas con "RESET-EN". También se puede deshabilitar el auto-reset conectando una resistencia de 110 Ω desde el pin 5V al pin de reset.

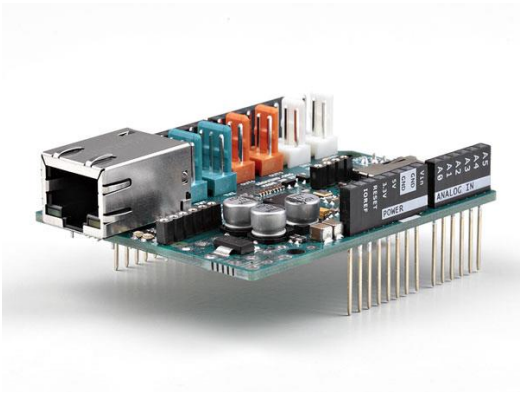
3.2.10 Características Físicas y Compatibilidad de Shields

La longitud y amplitud máxima de la placa Mega2560 es de 4 y 2.1 pulgadas respectivamente, con el conector USB y la conexión de alimentación sobresaliendo de estas dimensiones. Tres agujeros para fijación con tornillos permiten colocar la placa en superficies y cajas. Se debe tener en cuenta que la distancia entre los pines digitales 7 y 8 es 160 mil (0,16").

El Mega está diseñado para ser compatible con la mayoría de shields diseñados para el Uno, Diecimila o Duemilanove. Pines digitales 0 a 13 (y los pines AREF y GND adyacentes), las entradas analógicas de 0 a 5, los conectores de alimentación, y los conectores ICSP están todos ubicados en posiciones equivalentes. Además la UART principal (puerto serie) se encuentra en los mismos pines (0 y 1), al igual que las interrupciones externas 0 y 1 (pines 2 y 3, respectivamente). SPI está disponible a través de la cabecera ICSP tanto en el Mega2560 y Duemilanove / Diecimila.

3.3 Arduino Ethernet Shield 2





La Arduino Ethernet Shield 2 conecta Arduino a Internet en cuestión de minutos. Sólo es necesario conectar el shield a un Arduino, conectar éste a su red mediante el uso de un cable RJ45 y seguir unos pasos sencillos para empezar a controlar su mundo a través de internet. Como siempre con Arduino, todos los elementos de la plataforma - hardware, software y documentación - es de libre acceso y open source.

Descripción

La Arduino Ethernet Shield 2 permite a una placa Arduino conectarse a internet. Se basa en el chip Wiznet W5500 Ethernet. El Wiznet W5500 ofrece una red (IP) capaz de hacer uso de TCP y UDP. Soporta hasta ocho conexiones de socket simultáneas. Se hace uso de la biblioteca de Ethernet2 para escribir bocetos que se conectan a Internet a través del Shield. El

Ethernet Shield 2 se conecta a una placa Arduino mediante cabeceras wire-wrap las cuales se encuentran en la parte inferior del Shield. Esto mantiene la disposición de los pins intacta y permite la conexión de otros shields conectados sobre éste.

El Ethernet Shield 2 posee una conexión RJ-45 estándar, con un transformador de línea integrado y Power over Ethernet habilitado.

Posee una ranura para tarjetas micro-SD, que puede ser utilizada para almacenar archivos para su posterior utilización en la red. Es compatible con el Arduino Uno y Mega (utilizando la librería Ethernet2). El lector de tarjetas micro-SD del shield es accesible a través de la Biblioteca SD. Cuando se trabaja con esta biblioteca, SS se encuentra en el Pin 4.

Arduino se comunica tanto con el W5500 y la tarjeta SD mediante el bus SPI (a través de la cabecera ICSP). El pin 10 es utilizado para seleccionar el W5500 y el pin 4 para la tarjeta SD. Estos pines no se pueden utilizar para entradas o salidas. En los Arduino Mega, el pin SS, 53, no se utiliza para seleccionar el W5500 o la tarjeta SD, pero debe mantenerse como salida o la interfaz SPI.

Se debe tener en cuenta que debido a que el W5500 y la tarjeta SD comparten el mismo bus SPI, sólo uno puede estar activo. Si se hace uso de ambos periféricos, esto debe ser atendido por las bibliotecas correspondientes. Si no se hace uso de algunos de los periféricos, es necesario desactivarlos explícitamente. Para hacer esto con la tarjeta SD, se ajusta el pin 4 como salida y HIGH. Para el W5500, ajustar el pin digital 10 como salida y HIGH.

El shield contiene una serie de LEDs de información:

- ON: indica que la placa y el shield están alimentados
- LINK: indica la presencia de un enlace de red y parpadea cuando el shield transmite o recibe datos
- FDX: indica que la conexión de red es full duplex
- 100M: indica la presencia de una conexión de red de 100Mb/s (en contraposición a 10 Mb/s)

- ACT: parpadea cuando hay actividad RX o TX

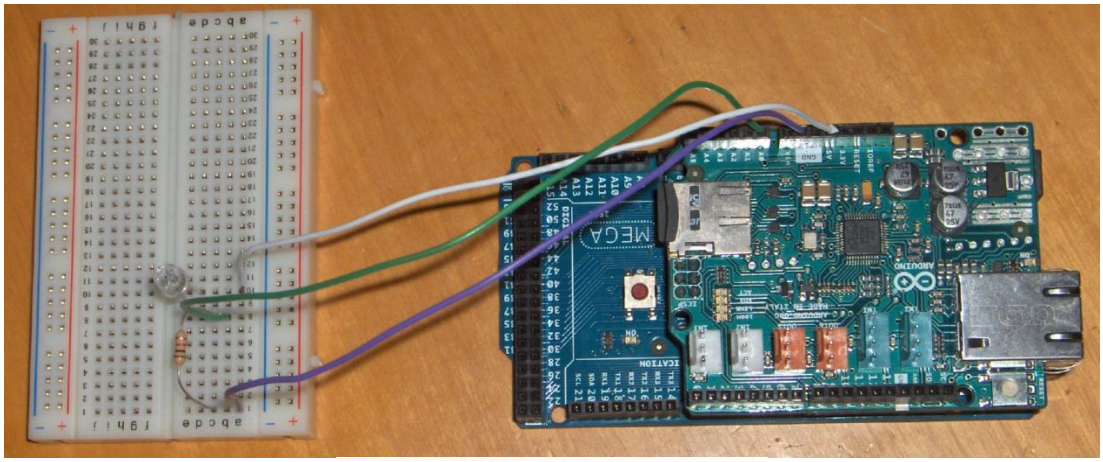
4 – Sistema Desarrollado

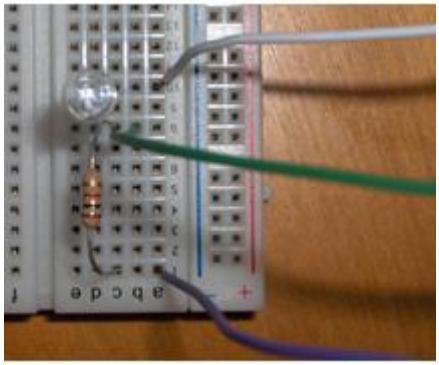
Para el correcto funcionamiento del servidor web que es necesario disponer de unos elementos básicos que son un PC con tarjeta Ethernet y una versión actualizada de su navegador web, junto con un Arduino que llevara conectados ciertos actuadores y sensores.

En nuestro navegador web podremos ver como se actualizan los valores de los sensores u actuadores y establecer un valor umbral con el que poder ser alertados en caso de que este sea sobrepasado. Esto puede ser útil, por ejemplo, en el caso de tener un sensor de calor y queremos que se nos avise en caso de sobrepasar cierta temperatura.

El funcionamiento del servidor web asíncrono implementado sobre Arduino explicado de una forma sencilla y sin entrar en detalles seguirá los siguientes pasos:

1. Establecimiento de la conexión entre servidor y cliente.
2. Envío de la página web al cliente.
3. Recepción de las peticiones del cliente
4. Tratamiento de las peticiones del cliente
5. Respuesta a las peticiones del cliente. En el caso de ser una petición GET, el cliente solicita la lectura de los sensores conectados al Arduino junto con los umbrales y recibirá los datos en un documento XML, sin ser necesaria la carga de toda la página web de nuevo. En el caso de la petición POST, el cliente habrá actualizado el umbral a un valor determinado por él y será enviado al servidor para que sea almacenado.





Para comenzar a programar el Arduino conectado, se han tenido en cuenta las dos partes principales de la programación en Arduino: la parte setup() en la que se configuraran los parámetros para la configuración de la placa, y la parte loop(), ejecutará la parte principal del programa.

Comencemos por explicar la parte setup() del código ya que sin ella el código no podría comenzar a funcionar.

Debido a que vamos a tener la página web en la tarjeta micro-SD debemos deshabilitar el chip Ethernet, por lo que el pin 10 se establece como salida. Se ha configurado la velocidad del bus serie a 9600, que transmitirá la información por tx0 y la recibirá por rx0, el hecho de emitir por estos pines lo tiene asignado la placa por hardware. Establecemos la MAC del shield ethernet y la IP de la red, una vez establecidos dichos parámetros arrancamos el servidor.

En la imagen inferior se puede observar cómo se hacen las comprobaciones para ver si la tarjeta es inicializada correctamente y si contiene el archivo que contendrá la página web.

```
void setup()
{
    // Deshabilitamos el chip Ethernet
    pinMode(10, OUTPUT);
    digitalWrite(10, HIGH);

    Serial.begin(9600);

    //Comprobacion de que la tarjeta ha sido inicializada

    Serial.println("Inicializando tarjeta SD...");
    if (!SD.begin(4)) {
        Serial.println("ERROR - En la inicializacion");
        return;
    }
    Serial.println("EXITO - Tarjeta SD inicializada.");
    //Comprobación de que archivo que contiene la pagina web
    //se encuentra en la tarjeta SD
    if (!SD.exists("index.htm")) {
        Serial.println("ERROR - No se encontro index.htm");
        return;
    }
    Serial.println("EXITO - Se encontro index.htm");

    Ethernet.begin(mac, ip);
    servidor.begin();
}
```

Prosigamos con la parte loop(). En ella se escucharan las llamadas de los clientes y se trataran las peticiones que realicen.

Si escuchamos la llamada de un cliente comenzaremos enviando la cabecera estándar de HTTP haciendo uso de la función cliente.println(), a través de la cual iremos enviando cada parte de la cabecera para que esta sea interpretada por nuestro navegador. Una vez ya enviada la cabecera pasamos a abrir el fichero que contiene nuestra página web que se encuentra en la tarjeta micro-SD, una vez que la página web ha sido abierta cerramos el documento.

```
// Envio de cabecera estandar
cliente.println("HTTP/1.1 200 OK");
cliente.println("Content-Type: text/html");
cliente.println("Connection: keep-alive");
cliente.println();
// Abrimos el fichero donde esta almacenada la página web
// y lo enviamos
Paginaweb = SD.open("index.htm");
if (Paginaweb) {
    while(Paginaweb.available()) {
        cliente.write(Paginaweb.read());
    }
    Paginaweb.close();
}
```

Una vez la página web ya está abierta en el navegador, volveremos al comienzo del loop. Es ahora cuando empezaremos a recibir peticiones por parte del cliente, que nos demandarán los valores de los sensores u otros dispositivos conectados a nuestro Arduino. Para poder analizar de manera más sencilla estos valores, se ha decidido enviar los datos con formato XML, por lo que la cabecera será diferente a la que enviamos para abrir la página.

```
// Envio de cabecera estandar
cliente.println("HTTP/1.1 200 OK");

//La siguiente parte de la cabecera es diferente dependiendo
//de si estamos usando XML o HTML
if (Busqueda_peti(petiHTTP, "val_sensor")) {
    // Enviamos del resto de la cabecera
    cliente.println("Content-Type: text/xml");
    cliente.println("Connection: keep-alive");
    cliente.println();
    // Llamada a la función que enviara los datos
    Funcion_XML_AJAX(cliente);
    Serial.println(val_alto);
}
```

Como se ve en la imagen superior se hace llamada a la función Funcion_XML_AJAX esta es la encargada de crear el documento XML que es enviado al cliente.

```
void Funcion_XML_AJAX(EthernetClient cli)
{
    int LDR_val;

    cli.print("<?xml version = \"1.0\" ?>");
    cli.print("<entradas>");

    LDR_val = analogRead(LDR_pin);
    cli.print("<analog0>");
    cli.print(LDR_val);
    cli.print("</analog0>");

    cli.print("<luminoso>");
    cli.print(val_alto);
    cli.print("</luminoso>");

    cli.print("</entradas>");
}
```

Expliquemos más detalladamente el significado de las etiquetas <analog0> y de <luminoso>. La primera de estas etiquetas, <analog0>, señala al valor del pin analógico 0, el cual, es leído y se almacena su valor en LDR_val cada vez que se recibe un GET. En el caso de que no estuviésemos usando un sensor de luz, podríamos tener que leer de unos de los pines digitales, aunque esto no afectaría a la creación de nuestro documento XML sería conveniente la creación de una etiqueta nueva indicando que corresponde a un pin digital. La segunda de las etiquetas, <luminoso>, señala al valor umbral enviado por el cliente mediante una petición POST, este valor umbral nos servirá para conocer en qué momento los datos arrojados por el sensor han sobrepasado el umbral establecido por el cliente.

Cabe destacar, que la mayor parte del tiempo que se ejecuta el servidor nos mantendremos en esta parte del bucle analizando los GET que recibimos, solo se cambiara en caso de recibir un POST o de desconexión del cliente.

Si del cliente lo que recibimos es un POST nos encargaremos de almacenar los datos que nos llegan desde el cliente y los analizamos, para poder dar una respuesta en consecuencia a su petición. Para este análisis, guardaremos toda la petición en un String y mediante el uso de la función indexof() buscaremos en el String una serie de caracteres, en nuestro caso “ALTO=”, y esta nos devolverá la posición en la que se encuentra nuestra cadena de texto dentro del String. Una vez localizada la posición podremos guardar los datos que vienen a continuación mediante el uso de la función substring(), la cual copiara la cadena hasta el final desde el lugar de inicio que le indiquemos. Por último convertimos la cadena que nos devuelve substring() en un entero para poder compararlo fácilmente con los valores que podemos obtener del sensor y volvemos a cargar la página web.

```
String rec;

while (cliente.available()) {
  rec += (char)cliente.read(); //Añade todo el mensaje
}
int txt = rec.indexOf("ALTO="); //busca esto
men = rec.substring(txt + 5); //aisla lo que hay despues
if (men != 0) { //Si has aislado un caracter no nulo
  Serial.println("");
  Serial.println(men); //Lo muestras por serial
  Serial.println("");
}

new_alto = men.toInt();
Serial.println(new_alto);
```

Una vez ya descrito el código que ejecuta el Arduino, pasemos ahora a comentar algunos aspectos importantes del documento "index.htm", el cual contiene la web.

El primer aspecto a comentar es CSS. La hoja de estilo está incorporada directamente en el documento HTML, a través del elemento style dentro de la sección head, consiguiendo de esta manera separar la información del estilo del código HTML.

Pasemos ahora a analizar la función más importante implementada dentro de <script>, comencemos por la función Sensor_Ajax(). La primera vez que se ejecuta simplemente haremos una petición mediante el uso de "GET" y volveremos al comienzo de la función. En esta ocasión crearemos una variable, denominada "request", que será la encargada de almacenar el objeto XMLHttpRequest. Para poder realizar acciones en base a la respuesta del servidor, comprobaremos los datos que nos llegarán del evento onreadystatechange, ya que es importante conocer los valores dados por readyState y status, debido a que la respuesta de la página será dada cuando readyState sea 4, indicándonos que la petición ha finalizado y la respuesta esta lista, y status sea iguala a 200, indicándonos que todo está correcto. Esta respuesta dependerá de los datos extraídos del documento XML, enviado por el servidor alojado en nuestro Arduino, ya que dependiendo de los valores que nos lleguen de este, abriremos una nueva ventana para mostrar de manera gráfica de que hemos llegado a uno de los límites marcados. Este límite es introducido por el usuario de la web rellenando el campo Umbral alto.

Mostremos ahora como queda el código de la función Sensor_Ajax().

```
function Sensor_Ajax()
{
    nocache = "&nocache=" + Math.random() * 1000000;
    var request = new XMLHttpRequest();
    var umbral0;
    var lud0;

    request.onreadystatechange = function()
    {

        if (this.readyState == 4) {
            if (this.status == 200) {
                if (this.responseXML != null) {
                    // Extraemos la información del documento XML
                    document.getElementById("entrada0").innerHTML =
                        this.responseXML.getElementsByTagName('analog0')[0].childNodes[0].nodeValue;
                    document.getElementById("entrada1").innerHTML =
                        this.responseXML.getElementsByTagName('luminoso')[0].childNodes[0].nodeValue;
                    umbral0=parseInt(this.responseXML.getElementsByTagName('luminoso')[0].childNodes[0].nodeValue);
                    lud0=parseInt(this.responseXML.getElementsByTagName('analog0')[0].childNodes[0].nodeValue);
                    if(umbral0<lud0){
                        ventana_secundaria=window.open("http://www.neoteo.com/wp-content/uploads/2013/07/A43C.jpg","LUZ1","width=316,height=288,left=200,menuba
                    }
                }
            }
        }
    }
    request.open("GET", "val_sensor" + nocache, true);
    request.send(null);
    setTimeout('Sensor_Ajax()', 1000);
}
```

5 - Conclusiones

En este Trabajo Fin de Grado se ha desarrollado un servidor web asíncrono para Arduino que es capaz de responder a peticiones GET y POST enviadas desde una página web que es ejecutada desde un navegador web. Estas peticiones provocan llamadas a los sensores y actuadores conectados al dispositivo pudiendo ir dirigidas directamente a recursos del Arduino en el que se ejecuta el servidor web.

Como primer paso para la realización del trabajo fin de grado, se ha creado un servidor AJAX sin el uso de la tarjeta micro-SD directamente en el Arduino. El cual enviaba el código HTML de la página línea a línea. Con el fin de que la página web fuera más gráfica y fuera más fácil de modificar, se decide pasar la página web a una tarjeta micro-SD, la cual está conectada al Arduino a través del Ethernet Shield. Esto nos ha permitido la inclusión de la hoja de estilo y la inclusión de otros elementos como ventanas secundarias.

El uso de XML en nuestro servidor hace que el número de sensores y actuadores que se conecten al Arduino no implique un gran aumento de líneas código, ni en el servidor, ni en la página web. Además es mucho más sencillo analizar los datos que son recibidos con este formato.

Se ha conseguido hacer uso de los actuadores y sensores conectados a los Arduinos según la petición que llega al servidor y que se le proporcione una respuesta correcta al cliente. Esto nos ha dado como resultado un servidor web asíncrono completamente operativo en Arduino, que es uno de los grandes objetivos por los que se ha llevado a cabo este trabajo fin de grado.

Al hacer uso de un servidor web asíncrono, se nos da la posibilidad de que al ampliar el código, las nuevas funciones que se añadan a la web podrán ejecutarse incluso cuando el servidor aún no haya devuelto la información que se le haya solicitado para estas.

Líneas de desarrollo futuro

La primera línea de desarrollo futura a llevar a cabo, es la de almacenar los datos de los sensores y los actuadores en un documento, indicando la hora a la que se produce cada medición, para así poder llevar un control más exhaustivo de los datos arrojados por estos.

Otra posible ampliación del trabajo sería la inclusión de una gráfica en la página web que se actualice en tiempo real. Lo que nos permitiría ver de manera más visual los cambios que se producen en los sensores, más que viendo simplemente los datos directamente arrojados por un sensor y visualizados en la web.

También cabe destacar, otra ampliación, en la cual aprovechando la asincronía de AJAX podríamos utilizar las propias peticiones de datos (GET), para determinar que solo deseamos recibir una respuesta del servidor cuando se cumplan los requisitos indicados en la petición, por ejemplo, en el caso de enviar una petición en la que indiquemos el umbral máximo y mínimo, el servidor no nos dará una respuesta hasta que no se detecte desde el sensor un valor que se encuentre entre ambos, mientras tanto el cliente puede seguir realizando otras funciones.

Bibliografía

http://arduino.cc/es/Main/ArduinoBoardMega

http://arduino.cc/en/Main/arduinoBoardMega2560

http://www.arduino.org/products/shields/5-arduino-shields/arduino-ethernet-shield-2

https://developer.mozilla.org/es/docs/AJAX/Primeros_Pasos

http://www.ibm.com/developerworks/ssa/library/wa-aj-webservices/

http://librosweb.es/libro/ajax/capitulo_1.html

http://definicion.de/html/

http://www.definicionabc.com/tecnologia/html.php

http://www.alegsa.com.ar/Dic/css.php

https://www.masadelante.com/faqs/css

http://www.maestrosdelweb.com/introcss/

http://definicion.de/xml/

http://www.hipertexto.info/documentos/xml.htm

http://www.mundolinux.info/que-es-xml.htm

http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML

http://www.w3schools.com/ajax/ajax_xmlhttprequest_onreadystatechange.asp

http://www.infor.uva.es/~jmrr/tgp/tgprecurso/intro1.htm

https://es.wikipedia.org/wiki/JavaScript

http://www.efectosjavascript.com/javascript.html

https://www.uv.es/jac/guia/jscript/javascr01.html