Treball de Fi de Grau
GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

CONTROL DE UNA CASA DOMOTICA VIA WEBSERVICE

Larry Telésforo Cárdenas Zavaleta

Director: José Bosch Estrada
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 28 de julio de 2017

Agradecimientos

A mi esposa, por su apoyo y ayuda incondicional.

A mis hijos, que son la alegría de mi vida.

A mi familia, que es la que siempre ha creído en mí.

A mis amigos, por sus ánimos constantes.

A mis compañeros de trabajo, por haber confiado en mí.

Y finalmente a mi tutor, por haberme brindado esta oportunidad.

Gracias a todos.

Resumen

Este proyecto consiste en la elaboración de un sistema domótico. Mediante el uso de un dispositivo tipo Arduino y un servidor, que dispone de una interface web donde se gestiona el control de la vivienda. Para demostrar el funcionamiento del sistema se ha realizado una maqueta de una vivienda, en la que se puede realizar simulaciones de los siguientes elementos del sistema domótico.

Abstract

This project consists of the development of a home automation system. By using a type of Arduino device and a server, it has a web interface where it manages the control of housing. To demonstrate the operation of the system has been made a model of a house, in which it is possible to make simulations of the following elements of the home automation system.

1. Indice

1. Introducción

En este primer capítulo comentaremos la motivación para la realización de este proyecto y los objetivos que se pretenden alcanzar.

1.1. Motivación

La informática es un campo muy amplio donde cada vez existen más tecnologías con diversos patrones y estándares. Por ello se hace necesario conseguir que todas las nuevas tecnologías se puedan comunicar entre ellas y de este modo lograr un objetivo de forma conjunta. El reto que la mayoría de informáticos tienen que abordar día a día.

El campo de la domótica ha despertado mi curiosidad, porque gracias a su tecnología y evolución conseguimos aumentar nuestro confort de vida en nuestras viviendas y facilitar nuestras tareas cotidianas.

El realizar este trabajo, supone enfrentarse a diferentes problemas que tenemos que hacer frente en cada capa y en la comunicación e interacción entre ellas.

1.2. Objetivo del proyecto

Este proyecto tiene como objetivo principal diseñar un sistema domótico para una casa inteligente. Y realizar un prototipo de este diseño mediante una maqueta en la que se representa una casa.

Se va a optar por ofrecer un sistema domótico con las siguientes funcionalidades.

Hardware:

- Controlar las 7 luces LED para la iluminación.
- Controlar 3 servos para mover puertas y persiana.
- Teclado control de acceso. (keypad).
- Alarma acústica (Buzzer).

- Alarma lumínica (Led Rojo).
- Ventilador para refrigeración de la casa.
- Reloj para tener la hora y fecha correcta en cada momento.
- Monitorizar sensores desde la web/app.
- Display 16x2 crystal liquid para monitorizar sensores sin entrar en la web/app.
- Detectar fuego, fuga de gas y notificar.
- Algoritmo control climático.
- Algoritmo control de luminosidad exterior.
- Contador de consumo eléctrico de la casa.
- Cámaras IP.
- Baterías.
- Captadores de Energía Solar y Eólica.
- Bomba de riego para jardín.
- Control remoto IR para dispositivos multimedia.

Software:

- Lógica interface Web/App.
- Compilar interface para Web, Android, IOS, Windows.
- Capa de interface responsiva.
- Base de datos.
- Acceso.
- Registro.
- Gestión de roles de usuario.
- Acceso parcial a las instalaciones de la Casa.
- Vista del plano de la casa con información en tiempo real.
- Número de Usuario conectados.
- Historial de Notificaciones.
- Gráficas de temperatura, humedad, Gas, etc.
- Chat interno entre usuarios de la casa.

- Calendario.
- Añadir eventos según los datos de los sensores.
- Despertador.

Para controlar todos estos aspectos se ha creado una interfaz web.

En el proyecto, se desarrollarán los siguientes módulos:

Arduino, se encargará del control físico de cada componente sensor.

Interfaz web que se desarrollará con HTML, CSS y Javascript que se encargará del control de la vivienda domótica, de la comunicación con Arduino y la comunicación con la interfaz web.

Una vez finalizado el proyecto se pretende haber conseguido varios aspectos:

Aplicar los diferentes conocimientos que se han adquirido a lo largo del Grado en Informática.

Trabajar con herramientas, técnicas, metodologías y tecnologías que se utilizan en los sistemas domóticos actualmente.

2. Lenguajes, tecnologías, componentes y herramientas necesarias

2.1. Lenguajes

Para desarrollar este proyecto se ha utilizado diferentes lenguajes de programación que explicaremos brevemente.

2.1.1. HTML

HTML (HyperText Markup Language) es un lenguaje de marcado para la elaboración de páginas web. Es un estándar a cargo de W3C, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. El lenguaje HTML basa su filosofía de desarrollo en la referenciación. Por ejemplo, para añadir un elemento externo a la página (imagen, vídeo, script, entre otros.), este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la página web contiene sólo texto mientras que recae en el navegador web (interpretador del código) la tarea de unir todos los elementos y visualizar la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma (estándar) por cualquier navegador web actualizado. [2]

2.1.2. CSS

CSS es el acrónimo de Cascading Style Sheets (hojas de estilo). Describe la apariencia y el formato de un documento con marcas (como HTML). CSS permite la separación del contenido (HTML) de su presentación. Maneja elementos como el diseño, colores, fuentes, etc. CSS permite asociar reglas

con los elementos que aparecen en un documento HTML. Estas reglas controlan cómo debe ser renderizado el contenido de dichos elementos. [2]

2.1.3. AngularJS

AngularJS es una librería para JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM (Document Object Model), manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Actualmente es la librería JavaScript más utilizada, es gratuita, de código abierto (bajo licencia MIT y GPL v2) y muy ligera. AngularJS, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. [2]

- AngularJS nos permite:
- Responder a una interacción del usuario.
- Acceder a elementos en un documento.
- Alterar el contenido de un documento.
- Recoger información del servidor sin refrescar la página.
- Modificar la apariencia de una web.
- Animar cambios en un documento.
- Aplicar nuevas vistas.

2.1.4. Arduino

Arduino se programa mediante el uso de un lenguaje propio basado en el lenguaje de programación de alto nivel C/C++. También es posible utilizar otros lenguajes de programación

ya que Arduino usa la transmisión serial de datos, la cual es soportada por la mayoría de los lenguajes, y para los que no soportan el formato serie de forma nativa, es posible utilizar software intermedio. [1]

2.2. Herramientas utilizadas para el desarrollo

Para desarrollar este proyecto se ha utilizado algunas herramientas de desarrollo para facilitar la programación. Se ha utilizado el IDE de arduino para programar y cargar el código dentro de la placa, Brackets que es un moderno editor de textos para los demás lenguajes utilizados, Git para el control de versiones y driver CH340 USB para cargar el programa a la placa Arduino.

2.2.1. IDE Arduino

Arduino ya cuenta con una plataforma de desarrollo en la cual podemos desarrollar el programa. Para ello nos tenemos que descargar de su página oficial la última versión del IDE Arduino. También necesitaremos la última versión de Java Runtime Enviroment (J2RE).

La interfaz del IDE de Arduino es muy sencilla. Tiene en la parte superior un menú´como la mayoría de programas en el que se pueden configurar diferentes aspectos como el puerto que vamos a usar y el tipo de placa Arduino.

Justo abajo tiene un botón  para compilar el código en busca de fallos y otro botón  para cargar el código en la placa Arduino.

En la parte central de la interfaz es donde se desarrolla el código y en la parte inferior la aplicación nos muestra diferente información cuando compilamos el código o lo cargamos a la placa, desde los posibles fallos hasta el espacio que ocupa en la placa Arduino.

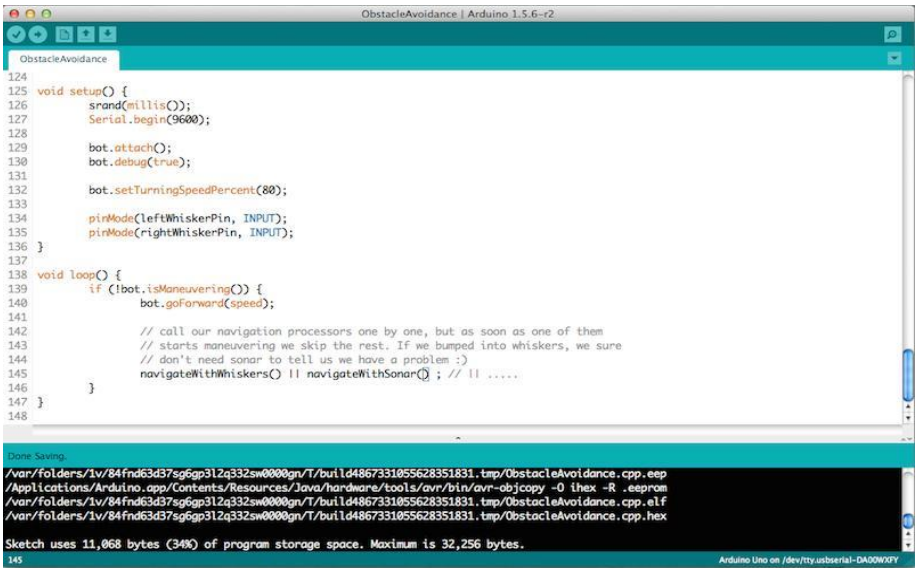

Imagen 1: IDE Arduino

2.2.2. Brackets

Impulsado por Adobe como parte de Adobe Edge, Brackets es un editor de texto construido sobre Tecnología Web para la Web. Brackets está enfocado en HTML, CSS y JavaScript.

El repositorio de Brackets en GitHub tiene actualmente 139 sucursales, 93 lanzamientos y 17165 compromisos a partir del 29 de enero de 2017. El código fuente está disponible gratuitamente bajo la licencia del MIT. Un desarrollador puede alterar las características de los soportes y personalizarlo para su propia conveniencia, bifurcando el código del software.

La edición rápida permite la edición en línea de los elementos CSS, Color Property y JavaScript para los desarrolladores. Esta función incorporada se puede aplicar a múltiples funciones o propiedades simultáneamente y todas las actualizaciones se aplican directamente al archivo asociado con los elementos modificados. [7]
Soportes proporciona varias características:

- Edición rápida
- Documentos rápidos
- Vista previa en vivo
- Fuente abierta
- Extensibilidad

Imagen 2: IDE Brackets

2.2.3. Framework Ionic

Ionic es un framework gratuito y open source para desarrollar aplicaciones híbridas multiplataforma que utiliza HTML5, CSS (generado por SASS), Angular, Node.js y Cordova como base. Es uno de los framework del momento por utilizar AngularJS para gestionar las aplicaciones, lo que asegura aplicaciones rápidas y escalables.

Una aplicación híbrida es aquella que permite desarrollar apps para móviles en base a las tecnologías web: HTML + CSS + Javascript. Son como cualquier otra aplicación de las que puedes instalar a través de las tiendas de aplicaciones para cada sistema, por lo que en principio usuarios finales no

percibirán la diferencia con respecto a las aplicaciones nativas. [6]

2.3. Tecnologías y formatos de datos utilizados

2.3.1. Ajax

Ajax es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad de las aplicaciones.

Decimos que Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (Scripting Language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté en formato XML.

Proporciona una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores ya que está basado en estándares abiertos como JavaScript y Document Object Model (DOM). [2]

2.3.2. REST

REST (Representational State Transfer) es un estilo de arquitectura de software para sistemas hipermedias distribuidos tales como la Web. El término fue introducido

en la tesis doctoral de Roy Fielding en 2000, quien es uno de los principales autores de la especificación de HTTP.
REST se refiere a una colección de principios para el diseño de arquitecturas en red. Estos principios resumen como los recursos son definidos y diseccionados. El término frecuentemente es utilizado en el sentido de describir a cualquier interfaz que transmite datos específicos de un domino sobre HTTP sin una capa adicional, como hace SOAP. Estos dos significados pueden chocar o incluso solaparse. Es posible diseñar un sistema software de gran tamaño de acuerdo con la arquitectura propuesta por Fielding sin utilizar HTTP o sin interactuar con la Web. Así como también es posible diseñar una simple interfaz XML+HTTP que no sigue los principios REST, y en cambio seguir un modelo RPC (Remote Procedure Call).
Cabe destacar que REST no es un estándar, ya que es tan solo un estilo de arquitectura. Aunque REST no es un estándar, está basado en estándares:

- HTTP
- URL
- Representación de los recursos: XML/HTML/GIF/JPEG/...
- Tipos MIME: text/xml, text/html, ...
- Los métodos HTTP más importantes son PUT, GET, POST y DELETE.

Como resultado, ni el cliente ni el servidor necesita mantener ningún estado en la comunicación. Cualquier estado mantenido por el servidor debe ser modelado como un recurso. [2]

2.3.3. XML

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de trozos de información. Ejemplos son un tema musical, que se compone de compases, que están formados a su vez por notas. Estas partes se llaman elementos, y se las señala mediante etiquetas.

Una etiqueta consiste en una marca hecha en el documento, que señala una porción de éste como un elemento. Un pedazo de información con un sentido claro y definido. Las etiquetas tienen la forma <nombre>, donde nombre es el nombre del elemento que se está señalando.

Ventajas del XML:

Es extensible: Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.

El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan bugs y se acelera el desarrollo de aplicaciones.

Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones. Podemos comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos, es decir, podríamos tener una aplicación en Linux con una base de datos Postgres y comunicarla con otra aplicación en Windows y Base de Datos MS-SQL Server.

Transformamos datos en información, pues se les añade un significado concreto y los asociamos a un contexto, con lo cual tenemos flexibilidad para estructurar documentos. [2]

2.4. Componentes Hardware

2.4.1. Arduino Mega 2560

El Arduino Mega 2560 es una placa con un microcontrolador Atmega2560. Cuenta con 54 pines entradas/salidas digitales (de los cuales 14 se pueden utilizar como salidas PWM) y 16 entradas analógicas, 4 UARTs (puertos de hardware de serie), un oscilador de cristal de 16 MHz, un puerto USB de conexión, un conector de alimentación, una cabecera de ICSP, y un botón de reinicio.
Contiene todo lo necesario para apoyar al microprocesador, sólo tiene que conectarlo a un ordenador con un cable USB, o con un alimentador adaptador AC-DC o una batería para comenzar. El Arduino Mega es compatible con la mayoría de los escudos diseñados para el Arduino.

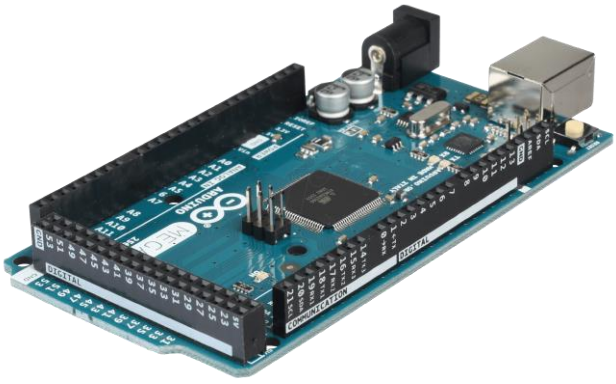

Imagen 3: Arduino Mega 2560

Especificaciones técnicas

Microcontrolador	ATmega2560
Tensión de funcionamiento	5V
Tensión de entrada (recomendado)	7-12V
Tensión de entrada (límite)	6-20 V
Digital I / O Pins	54 (de los cuales 15 proporcionan una salida PWM)
Analog Input Pins	dieciséis
Corriente CC por pin de E / S	20 mA
Corriente de CC para el Pin de 3.3V	50 mA
Memoria flash	256 KB de los cuales 8 KB utilizados por bootloader
SRAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz
LED_BUILTIN	13
Longitud	101,52 mm
Anchura	53,3 mm
Peso	37 g

Imagen 4: Características técnicas Arduino Mega 2560

2.4.2. Sensores

2.4.2.1. Temperatura y humedad

El sensor DHT11 que nos permite medir la temperatura y humedad con Arduino. Una de las ventajas que nos ofrece el DHT11, además de medir la temperatura y la humedad, es que es digital. El mayor inconveniente que podemos encontrar es su poca precisión.

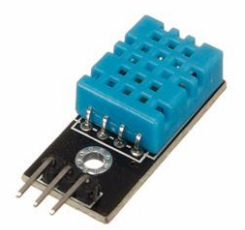

Imagen 5: Sensor DHT11

MODELO	DHT11
Alimentación	de 3,5 V a 5 V
Consumo	2,5 mA
Señal de salida	Digital
Temperatura	
Rango	de 0ºC a 50ºC
Precisión	a 25ºC ± 2ºC
Resolución	1ºC (8-bit)
Humedad	
Rango	de 20% RH a 90% RH
Precisión	entre 0ºC y 50ºC ± 5% RH
Resolución	1% RH

Imagen 6: Características técnicas DHT11

2.4.2.2. Gas

El MQ-135 puede detectar concentraciones humo y gas combustible de 20 a 2000 ppm (partes por millón). Este sensor tiene una alta sensibilidad y un tiempo de respuesta rápido. La salida del sensor es una resistencia análoga. El circuito de interfaz es muy simple, todo lo que se necesita hacer es alimentarlo con 5V, añadir una resistencia de carga y conectar la salida al conversor análogo – digital.

Imagen 7: Sensor MQ135

TECHNICAL DATA **MQ-135 GAS SENSOR**

FEATURES

Wide detecting scope — Fast response and High sensitivity
Stable and long life — Simple drive circuit

APPLICATION

They are used in air quality control equipments for buildings/offices, are suitable for detecting of NH3,NOx, alcohol, Benzene, smoke,CO_2 ,etc.

SPECIFICATIONS

A. Standard work condition

Symbol	Parameter name	Technical condition	Remarks
Vc	Circuit voltage	5V±0.1	AC OR DC
V_H	Heating voltage	5V±0.1	ACOR DC
R_L	Load resistance	can adjust	
R_H	Heater resistance	33Ω ±5%	Room Tem
P_H	Heating consumption	less than 800mw	

B. Environment condition

Symbol	Parameter name	Technical condition	Remarks
Tao	Using Tem	-10℃...+45℃	
Tas	Storage Tem	-20℃...+70℃	
R_H	Related humidity	less than 95%Rh	
O_2	Oxygen concentration	21%(standard condition)Oxygen concentration can affect sensitivity	minimum value is over 2%

C. Sensitivity characteristic

Symbol	Parameter name	Technical parameter	Remark 2
Rs	Sensing Resistance	30KΩ -200KΩ (100ppm NH_3)	Detecting concentration scope： 10ppm-300ppm NH_3 10ppm-1000ppm Benzene 10ppm-300ppm Alcohol
α (200/50) NH_3	Concentration Slope rate	≤ 0.65	
Standard Detecting Condition	Temp: 20℃±2℃ Humidity: 65%±5%	Vc:5V±0.1 Vh: 5V±0.1	
Preheat time	Over 24 hour		

Imagen 8: Características técnicas MQ135

2.4.2.3. Luminosidad

Para el control de la luminosidad se utiliza una LDR (Light-Dependent Resistor). Una LDR o fotorresistor es un componente electrónico cuya resistencia varía en función de la luz. Utilizamos una entrada analógica en Arduino para comprobar su valor y ver la luminosidad actual.

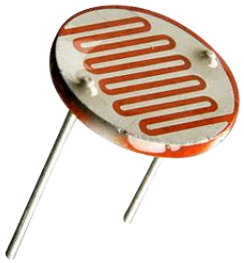

Imagen 9: LDR

2.4.2.4. Llama Fuego

Un sensor de llama óptico es un dispositivo que permite detectar la existencia de combustión por la luz emitida por la misma. Esta luz puede ser detectada por un sensor óptico, y ser capturado por las entradas digitales y las entradas analógicas de Arduino.

Imagen 10: Llama

2.4.3. Pantalla LCD

El LCD (Liquid Crystal Display) o pantalla de cristal líquido es un dispositivo empleado para la visualización de contenidos o información de una forma gráfica, mediante caracteres, símbolos o pequeños dibujos dependiendo del

modelo. Está gobernado por un microcontrolador el cual dirige todo su funcionamiento.
En este caso vamos a emplear un LCD de 16x2, esto quiere decir que dispone de 2 filas de 16 caracteres cada una. Los píxeles de cada símbolo o carácter, varían en función de cada modelo.
En nuestro proyecto mostramos la temperatura y humedad (exterior/interior) y la hora local fecha.

Imagen 11: Liquid Crystal Display

2.4.4. Relés

Un relé, de modo sencillo, es un interruptor accionado electrónicamente y se ubica en un circuito eléctrico para interrumpir o no el flujo de corriente eléctrica, según un circuito de control.
Cuando se alimenta el circuito, el relé cierra permitiendo el flujo de corriente entre los dos puntos conectados. Cuando se retira el comando de control, o sea, la energía, se abre el circuito.

Imagen 12: Relés

2.4.5. RTC - DS1307

El DS1307 de Maxim/Dallas es un circuito integrado capaz de almacenar y llevar la cuenta de la fecha y hora, además disponemos de unos cuantos bytes de datos de usuario en su memoria RAM no volátil (NVRAM).

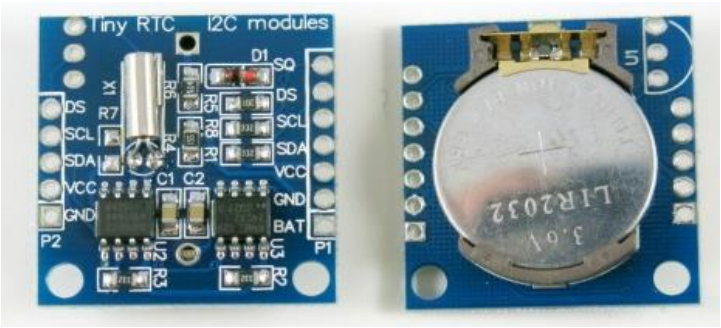

Imagen 13: RTC - DS1307

3. Análisis de una vivienda inteligente con Arduino

En este capítulo se ha realizado un análisis de los requisitos que cumplirá la casa y los problemas que se tiene que tratar al ser un sistema a tiempo real.

3.1. Descripción del Caso de Estudio

Para el Trabajo Final de Grado he diseñado una casa inteligente.
El objetivo es controlar las luces, saber la temperatura y humedad desde cualquier dispositivo. Para ello se ha creado la interfaz web para poder interactuar con la vivienda en la que más adelante se pueda añadir más funcionalidades.
La luz de la casa se tiene que encender/apagar desde la interfaz web, de esta manera, evitamos los interruptores.
Uno de los requisitos es saber la temperatura y humedad (interior/exterior) de la casa, esta información se muestra en la web, app y a través de la pantalla LCD.

3.2. Análisis del Sistema Requerido

El principal problema dentro de una vivienda es que se trata de datos a tiempo real, el cual está en constante cambio, del que no se tiene un control absoluto. Ya que en el mundo real no se puede controlar lo que se desea.
En un sistema de una casa inteligente ocurren constantemente eventos procedentes del exterior el cual influye cualquier tipo de dispositivo, o desde el propio sistema para el control mediante los diferentes actuadores. Para almacenar y procesar toda esta información hemos decidido abordar el problema utilizando bases de datos y socket ya que se trata de un sistema en tiempo real.

Haré una representación digital del exterior. Para ello se hará uso de Arduino, el cual nos proporciona la abstracción ya que se encarga de controlar todos los actuadores y sensores.
Por último, tenemos una interfaz web que se encargará de mostrar el estado de la vivienda, y con el cual se puede interactuar para encender o apagar puntos de iluminación abrir y cerrar puertas.

3.3. Capa física

La capa física la componen todos los dispositivos físicos de la casa, entre los que se encuentra la placa Arduino, que es el encargado de controlar directa o indirectamente al resto. Arduino es el encargado de convertir el mundo físico al mundo digital.
El objetivo es el control de la temperatura y la humedad mediante diferentes sensores tanto en el exterior/interior de la casa. Se dispone de unos sensores de llama en la cocina por si hubiera un posible incendio y un sensor de gas por si hubiese una fuga. En el exterior controlaremos la luminosidad para que se abra una ventana. La vivienda dispone de bombillas led que están conectados a unos relés para controlarla de forma remota mediante automatismos.
La vivienda también dispone de una pantalla LCD para el control que nos muestra la temperatura/humedad del exterior/interior además de la hora local y fecha.

3.4. Capa control

La capa de control está formada por Arduino. Este procesa cada estado de la vivienda domótica almacenando su valor. En nuestro caso al tratarse de una vivienda autónoma, no se

realizará ningún tipo de acción sobre ella. Aun así, esta parte de capa de control es imprescindible para tener una representación digital de la vivienda y posteriormente poder transmitir esa información a la capa de presentación.

3.5. Capa de presentación

Para la creación de la interfaz web realizó en primer lugar un diseño el cual esté enfocado dependiendo del uso y las necesidades del usuario final. Para lograr esto habría que recolectar información de los usuarios y sus necesidades. En segundo lugar, hay que crear el diseño de la interfaz, y hacer una evaluación del diseño de dicha interfaz, y en caso de no satisfacer los criterios establecidos previamente volver al realizar el diseño hasta que los requisitos se cumplan. Por último, se implementaría y se evaluaría para comprobar que cumple todos los criterios de usabilidad y requisitos establecidos, y si no es así volver a diseñar la interfaz y repetir todos los pasos.

En este caso simplemente vamos a crear una sencilla interfaz con el único objetivo de que tenga la funcionalidad necesaria para poder interactuar con nuestra vivienda inteligente.

Para la construcción de la interfaz web y app se ha se ha utilizado el framework ionic.

Un aspecto importante en el diseño de la interfaz web es como tener actualizada la pantalla del usuario, el estado actual de la vivienda domótica. Principalmente los estados de iluminación, puertas, persiana, temperatura, humedad, fecha y hora.

3.6. Comunicación entre capa física y capa de control

Con respecto a la parte física, la comunicación entre Arduino y la interfaz web se ha realizado por Ethernet que a su vez está conectado a red doméstica.
Para la interacción de mensajes entre ambas capas hemos decidido crear un lenguaje propio para que ambas capas puedan mandar y recibir mensajes ya que Arduino no proporciona ningún lenguaje predefinido para mandar los estados de los dispositivos físicos.

4. Diseño

Se ha elaborado una maqueta de madera a escala simulando la casa.

4.1. La Casa

Se ha diseñado a escala una casa que consta de una sala, tres habitaciones, una cocina, un baño y un garaje. El cual mostramos en la imagen.

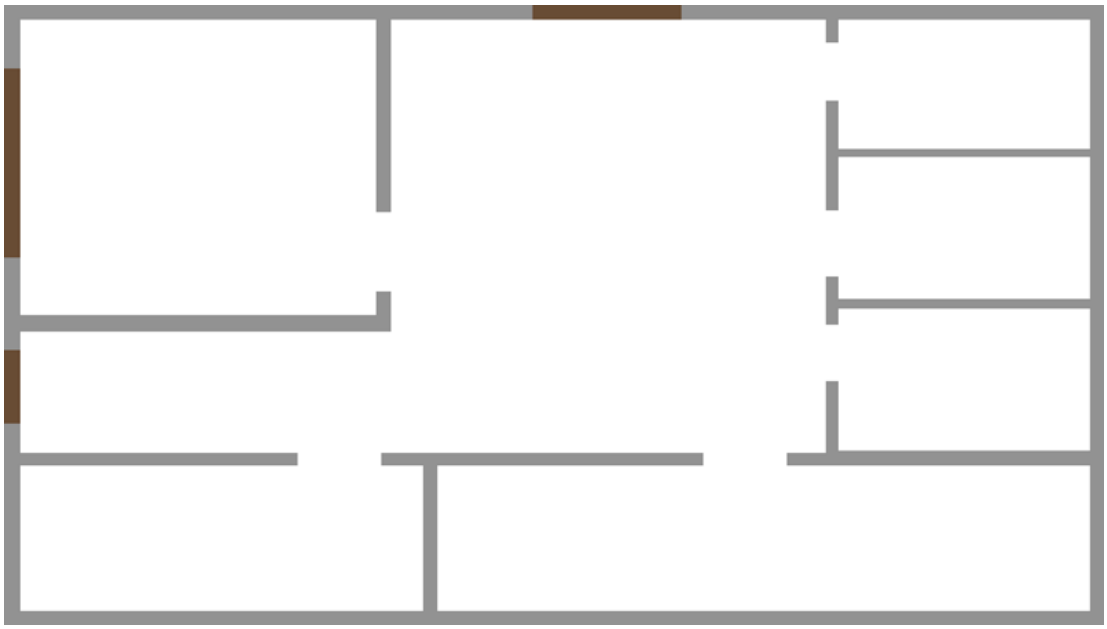

Imagen 14: Plano de la vivienda

Imagen 15: Maqueta inicial

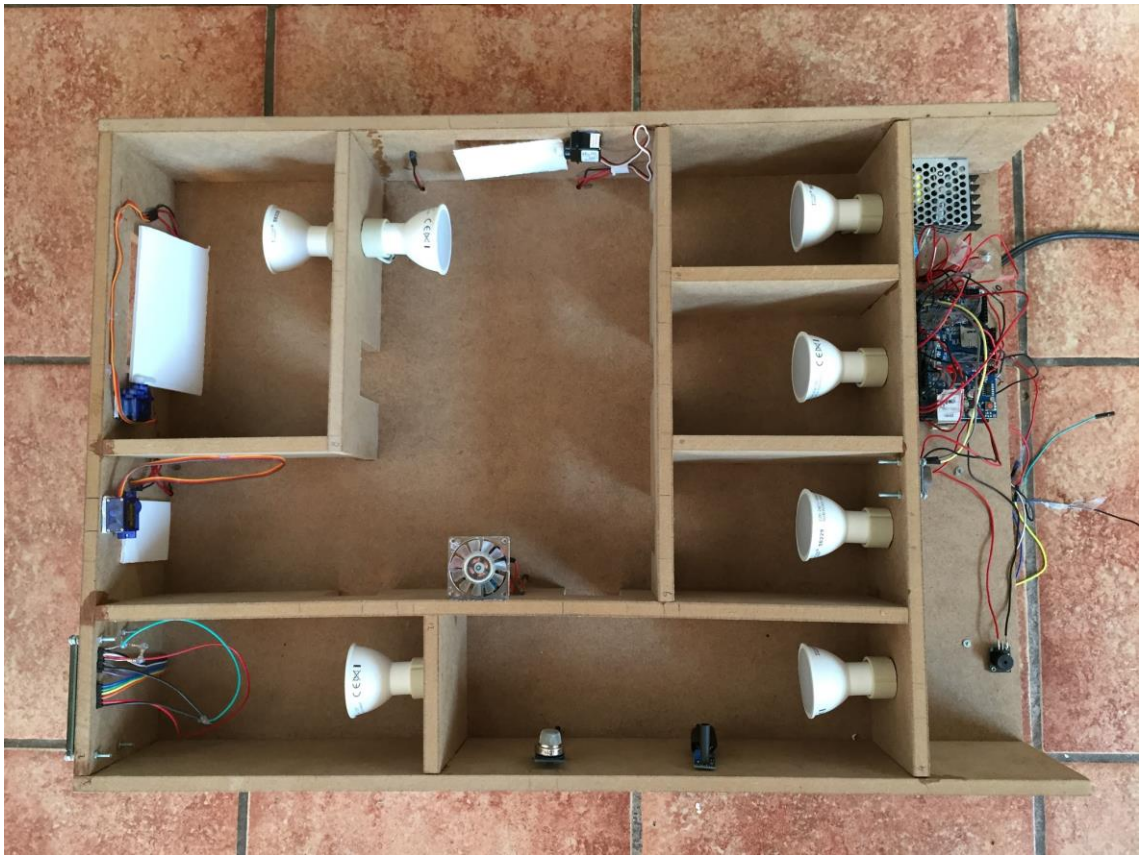

Imagen 16: Maqueta final

4.1.1. Sensores y actuadores

Existen dos sensores de temperatura y humedad situados en la sala y en el exterior. Cada habitación de la casa, tiene un punto de iluminación controlado por siete relés, conectados al sistema de iluminación. Hay tres servos que están situados en la puerta principal, garaje y ventana de la sala. En la cocina tenemos dos sensores; de llama, y de gas. Nos avisan de un posible incendio o fuga. Hay un ventilador en la sala que hace la función de aire acondicionado que se activa dependiendo de la temperatura ya configurada.
Para terminar, tenemos una pantalla LCD que nos mantiene informados de la temperatura y humedad (interior/exterior) sin necesidad de entrar a la web o app.
Permitiendo que Arduino tenga el control de cada punto.

Imagen 17: Componentes de la casa 1

4.2. Arduino

La capa física será controlada por Arduino, el cual nos permite convertir el mundo físico en digital. Cada sensor o actuador tiene su propia librería en Arduino que se encarga de controlarlos, permitiendo centrarnos en recolectar la información digital que nos proporciona y mandar dicha información cuando sea necesario a la interfaz web o app.

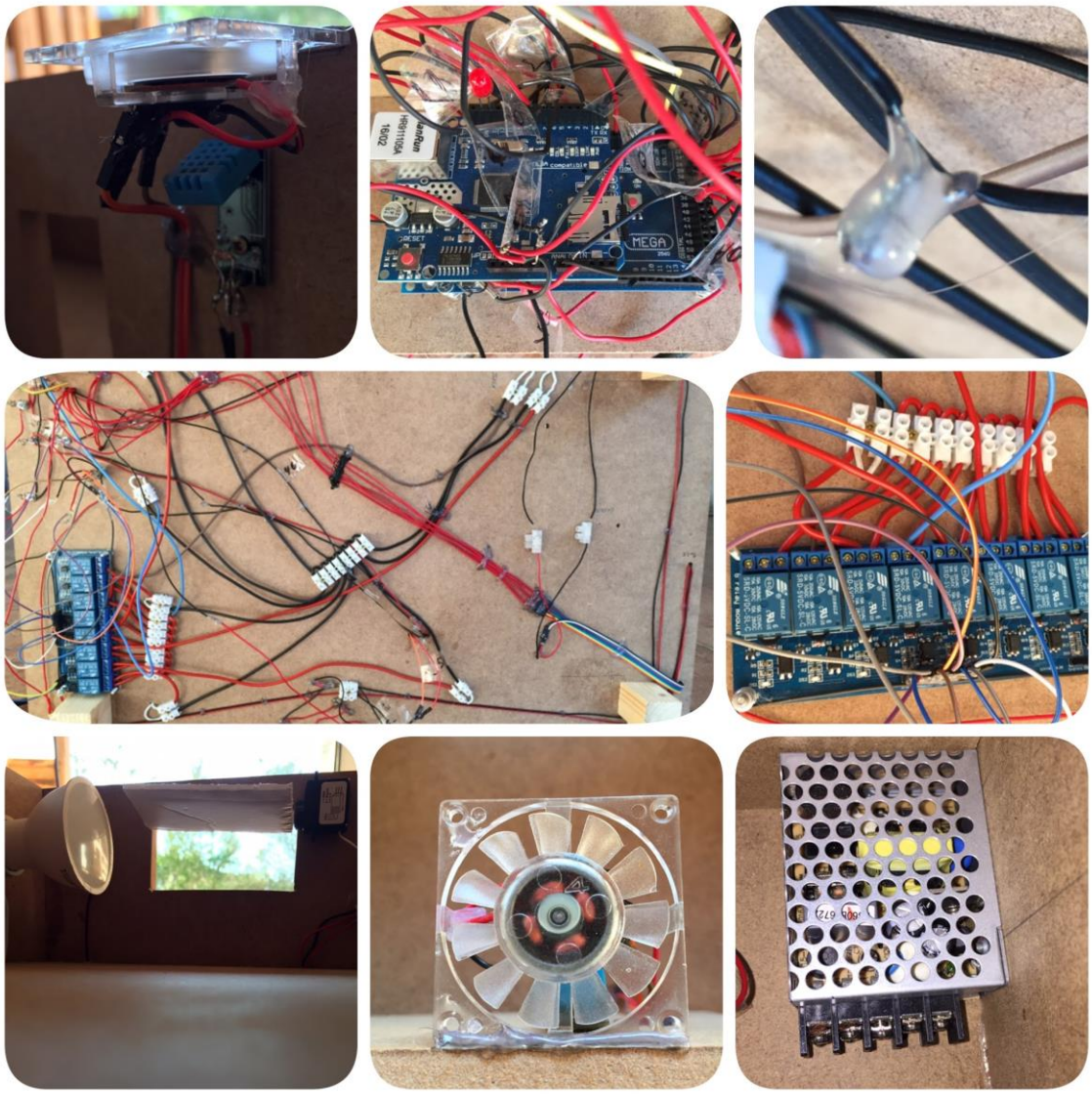

Imagen 18: Componentes de la casa 2

4.2.1. Distribución de pines

Esta es nuestra tabla donde especificamos cada pin de la placa Arduino y para que se ha utilizado, si es digital o analógico, y si hace función de entrada o de salida.

N° Pin	Función	Digital/Analógico	Entrada Salida	Casa
A0	MQ135	Analógico	Entrada	Cocina
A1	LDR	Analógico	Entrada	Exterior
A2	Buzzer	Analógico	Salida	Exterior
2	DHT 1	Digital	Entrada	Salón
18	DHT 2	Digital	Entrada	Exterior
46	Ventilador	Digital	Salida	Salón
13	Led Notify	Digital	Salida	Exterior
20	Reloj	Digital	SDA	Exterior
21	Reloj	Digital	SCL	Exterior
28	Servo P1	Digital	Salida	Puerta Entrada
30	Servo P2	Digital	Salida	Puerta Garaje
32	Servo P3	Digital	Salida	Persiana Salón
24	Relé 1	Digital	Nulo	Nulo
11	Relé 2	Digital	Salida	Luz Baño
23	Relé 3	Digital	Salida	Luz Garaje
9	Relé 4	Digital	Salida	Luz Salón
8	Relé 5	Digital	Salida	Luz Cocina
7	Relé 6	Digital	Salida	Luz Cuarto 1
6	Relé 7	Digital	Salida	Luz Cuarto 2
25	Relé 8	Digital	Salida	Luz Cuarto 3
14	Flama	Digital	Entrada	Cocina
31	LCD	Digital	Salida	General
33	LCD	Digital	Salida	General
35	LCD	Digital	Salida	General
37	LCD	Digital	Salida	General
39	LCD	Digital	Salida	General
41	LCD	Digital	Salida	General

Tabla 1: Distribucion de pines

4.3. Interfaz web

Para la interfaz web, como he comentado anteriormente en el apartado de análisis, voy a realizar un diseño muy básico, en el que se muestre la temperatura, humedad, gas, incendio, fecha, hora y estado de la iluminación, puertas y ventana. La única interacción que va a existir es apagar y encender cada punto de iluminación, el aire acondicionado (ventilador) se enciende automáticamente al igual que la persiana.

Únicamente hay una pantalla en el que se muestra toda la información. Hay unos botones que controlan la iluminación, las puertas y persiana.

Con respecto a la parte dinámica la cual gestionará la App, podemos diferenciar en dos partes. La primera de ellas es cuando el usuario presione el botón ON/OFF o ABRIR/CERRAR. La segunda parte trata sobre la actualización automática que he realizado en el estado de la vivienda, es decir, en la iluminación, gas, llamas, fecha, hora, temperatura y humedad.

Cuando el usuario presione ON/OFF se activa en el método correspondiente el cual realiza una petición REST. Posteriormente cambia el estado de la iluminación ENCENDIDO/APAGADO. Es importante saber que la petición REST tiene una respuesta de "OK" si la petición se ha realizado correctamente, es decir, si le ha llegado la petición a Arduino. [5] [6]

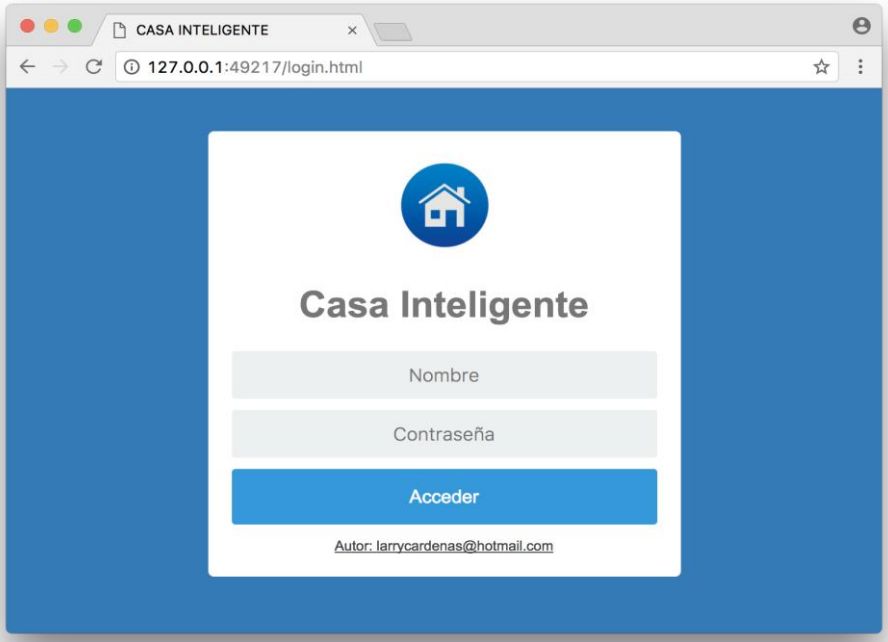

Imagen 19: Pantalla principal o Login

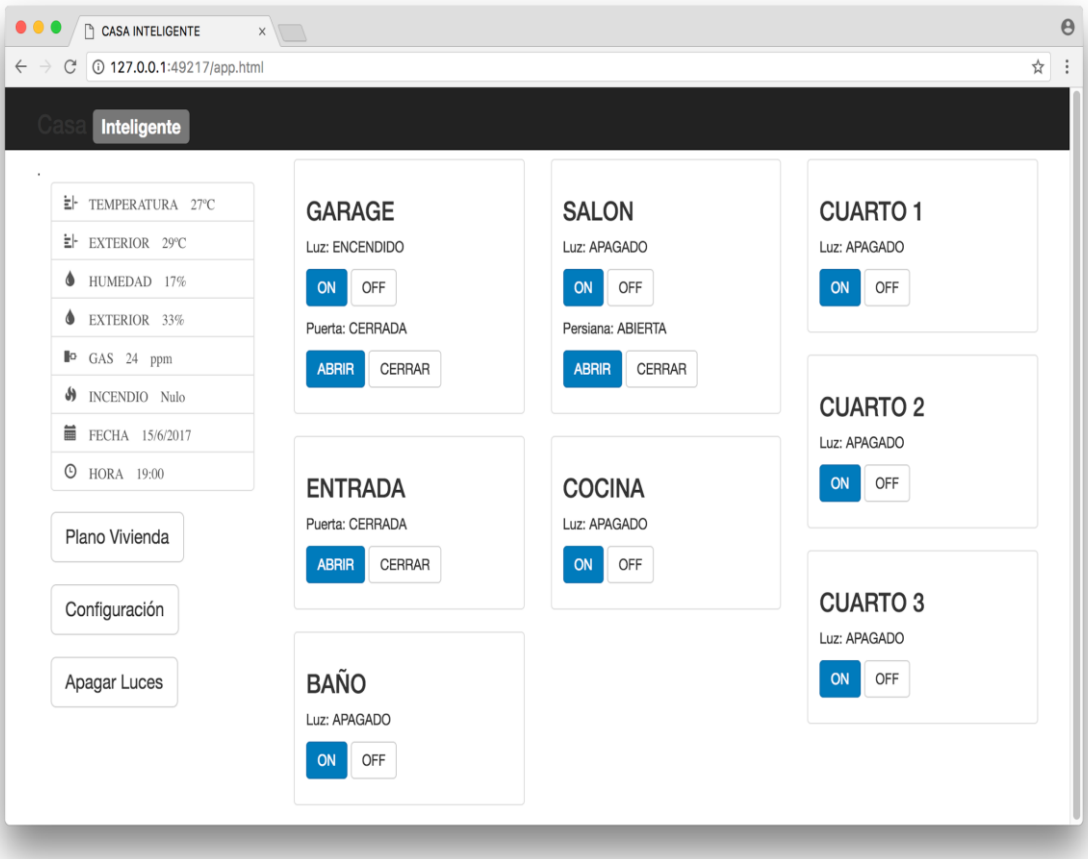

Imagen 20: Panel de control o Monitorización

5. Implementación

La implementación la hemos dividido en dos bloques principales. El primero de ellos es Arduino, el segundo es la interfaz web desarrollada en HTML con la tecnología cordova y el framework ionic para aplicación elaborar la aplicación móvil.

5.1. Instalación del Framework Ionic

Para poder utilizar el Framework Ionic necesitaremos disponer de NodeJS instalado en nuestro equipo. Una vez instalado, utilizaremos el gestor de paquetes NPM para instalar Ionic y Cordova utilizando el siguiente comando:
sudo npm install -g cordova ionic

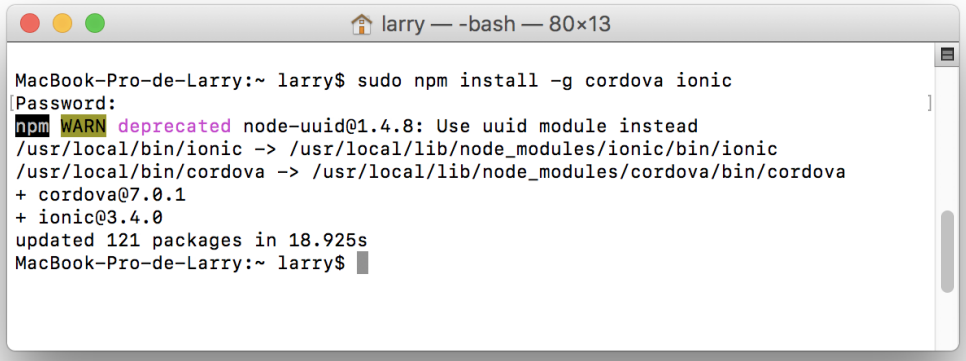

Imagen 21: Instalación del Framework Ionic

Dado que NodeJS [4] se ha convertido en una herramienta esencial para desarrolladores, es muy posible que ya lo tengas instalado, ya que mucho de los complementos de desarrollo frontend trabajan con NodeJS. La mejor manera de saber si ya tenemos Node instalado es lanzar el comando en la consola:

node -v

Y si está instalado nos debería informar la versión que tenemos de esta manera:

v8.1.2

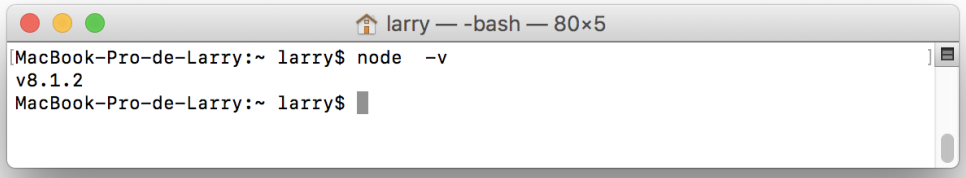

Imagen 22: Comprobar version de NodeJS

Para la creación de un nuevo proyecto, nos dirigimos a un directorio (previamente creado), y ejecutamos el siguiente comando para crear la estructura y los ficheros propios de un proyecto de Cordova:

Ionic start casainteligente blank

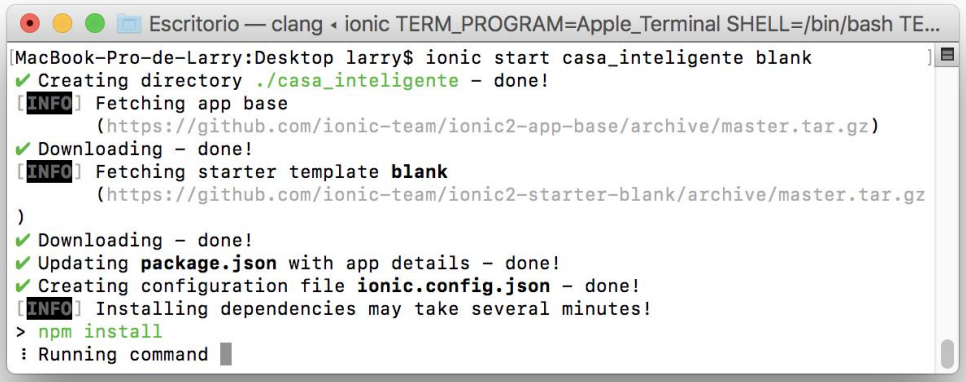

Imagen 23: Creación de un proyecto nuevo

Por defecto, Cordova [5] no trae añadida ninguna plataforma sobre la cual se compilará nuestra aplicación. Para añadir una, utilizamos el siguiente comando:
ionic cordova platform add [amazon-fireos | android | blackberry10 | browser | firefoxos | ios | osx | webos]

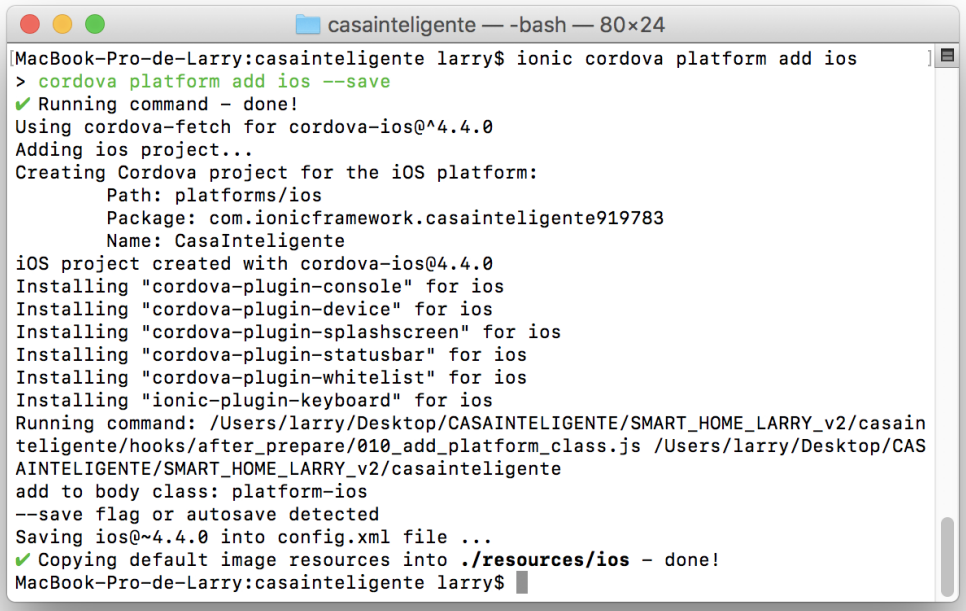

Imagen 24: Compilación para la plataforma IOS

Como podemos observar, Cordova nos ofrece un amplio repertorio de plataformas, por lo que conseguiremos hacer funcionar nuestra aplicación en casi cualquier sistema operativo disponible en el mercado. En caso de querer conocer cuáles son las plataformas que tenemos instaladas en nuestro proyecto, podemos utilizar el siguiente comando:
cordova platform ls

El directorio por defecto de Cordova para alojar los archivos del proyecto es '[raíz del proyecto]/www'. Aquí dentro deberemos colocar los archivos HTML, CSS y JS de nuestro proyecto web. Por ejemplo, podemos utilizar los archivos que Ionic Creator genera cuando descargamos el código fuente de nuestra aplicación. Todo lo que se encuentre dentro del directorio 'www' será encapsulado por Cordova, y el contenido será mostrado cuando la aplicación se ejecute en un dispositivo.

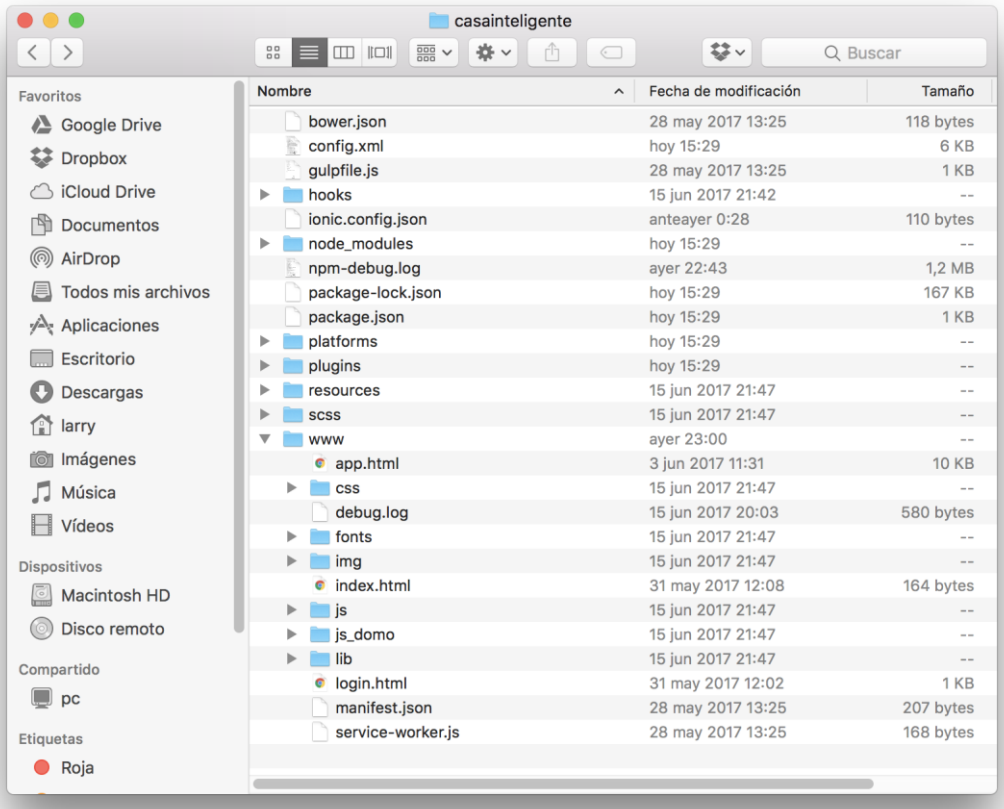

Imagen 25: Estructura del contenido del proyecto

Según el sistema operativo de destino, deberemos tener en cuenta unos ciertos requerimientos. Para poder compilar la aplicación para iOS es necesario disponer de un equipo con MacOS, ya que no es posible compilar un proyecto para iPhone o iPad sin Xcode. En el caso de Android, necesitaremos tener instalado el IDE oficial (Android Studio), que se encuentra disponible para plataformas MacOS, Linux y Windows.
Una vez cumplidos los requerimientos anteriores, utilizaremos el siguiente comando para compilar la aplicación:

cordova build [android | ios]

La utilidad nos generará un tipo de archivo que variará según la plataforma. En el caso de Android, la salida resultante será un CasaInteligente.apk, que podremos instalar de forma directa en cualquier dispositivo de esta plataforma.
En el caso de iOS se generará un fichero.xcodeproj, que podremos abrir en el propio Xcode. Desde Xcode tendremos la posibilidad de transferirlo a nuestro dispositivo iOS y testear allí la aplicación.
Aun así, en el caso de no disponer de un dispositivo iOS o Android, podemos utilizar los emuladores de cada una de las plataformas para visualizar la aplicación en nuestro equipo de igual forma que se vería en un dispositivo. Para ello, utilizaremos el siguiente comando:

cordova emulate [android | ios]

5.2. Arduino

A continuación, describimos los pasos que hemos realizado para la implementación.

5.2.1. Configuración IDE Arduino

En primer lugar, lo que tenemos que hacer cuando vamos a desarrollar un programa en Arduino es indicarle que placa vamos a utilizar, ya que sin esto luego no se puede compilar y cargar el código en la placa Arduino. En nuestro caso como utilizamos la placa Mega2560 hacemos lo siguiente: *Herramientas -> Placa -> Arduino Mega o Mega 2560*. [1]

5.2.2. Importar librerías

Como ya ocurre con otros lenguajes como Java, en Arduino también se incluyen las librerías que vayamos a utilizar al inicio del programa.
Para importar la clase que necesitemos tenemos que añadirla al proyecto de Arduino donde estamos trabajando.
Desde la interfaz de Arduino podemos añadir la librería situándonos en Programa -> Importar Librería -> Anadir librería.
Por último, en el programa solo tendremos que añadir los nombres de las librerías. En nuestro programa usaremos las siguientes librerías:

```
#include <Wire.h>  //reloj ++
#include <RTClib.h> //reloj
#include <SPI.h> // SD ++
#include <Ethernet.h> // Servidor WEB
#include <SD.h> // SD
#include <DHT.h> //Sensor luminosidad
#include <Servo.h> //motores
#include <LiquidCrystal.h> //pantalla
```

5.2.3. Declaración de Variables

```
#define FAN "FAN.txt"
#define R1 "R1.txt"
#define R2 "R2.txt"
#define R3 "R3.txt"
#define R4 "R4.txt"
#define R5 "R5.txt"
#define R6 "R6.txt"
#define R7 "R7.txt"
#define R8 "R8.txt"
#define P1 "P1.txt"
#define P2 "P2.txt"
#define P3 "P3.txt"
#define P4 "P4.txt"
#define TEMP1 "1.txt"
#define TEMP2 "2.txt"
#define TERMO1 "3.txt"
#define TERMO2 "4.txt"
RTC_DS1307 rtc; //reloj
DHT dht(2, 11);
DHT dht2(18, 11);
Servo puerta1;
Servo puerta2;
Servo puerta3;
LiquidCrystal lcd(31, 33, 35, 37, 39, 41);
```

5.2.4. Configuración inicial

La configuración inicial se realiza en la función setup(). Esta función es necesaria que se encuentre en el programa, incluso si no se tuviera que realizar ninguna configuración inicial. En nuestro caso, sí que tenemos que hacerla, la cual detallamos a continuación:

Inicializar comunicación serie:

```
Serial.begin(115200);
```

Inicializamos el sensor de temperatura y de humedad:

```
dht.begin();
dht2.begin();
Inicializar la pantalla LCD:
lcd.begin(16, 2);
lcd.createChar(0, grado_ico);
lcd.createChar(1, hum_ico);

Definir el uso de los pines digitales:
pinMode(2, INPUT);
pinMode(18, INPUT);
pinMode(ledPin, OUTPUT);
pinMode(46, OUTPUT);
pinMode(inputPin, INPUT);
pinMode(DOUTpin, INPUT);
pinMode(pinSpeaker, OUTPUT);
puerta1.attach(puerta[1]);
puerta2.attach(puerta[2]);
puerta3.attach(puerta[3]);
Inicializamos los relés:
for(byte a = 1 ; a<=8; a++)
{
    pinMode(RR[a], OUTPUT);
    digitalWrite(RR[a], offf);
}
```

5.2.5. Bucle principal

El bucle principal en Arduino es la función loop(). Esta se ejecuta después de la función setup() y la inicialización de variables. Esta función se ejecuta de forma indefinida, con lo cual es donde se llama a todos los eventos o procesos que se encargará de regular y gestionar todo el sistema.

En nuestro caso la función loop() es la encargada de llamar a los siguientes métodos:

```
pantalla();
lcd_casa();
reloj();
humedad();
mq7();
ldr ();
flame();
rele_iluminacion();
```

5.2.6. Sensor Gas MQ135

Con la función MQ135() obtenemos el valor del sensor y activamos la alarma sonora y acústica mediante un LED y un Buzzer si superar los 600PPM.

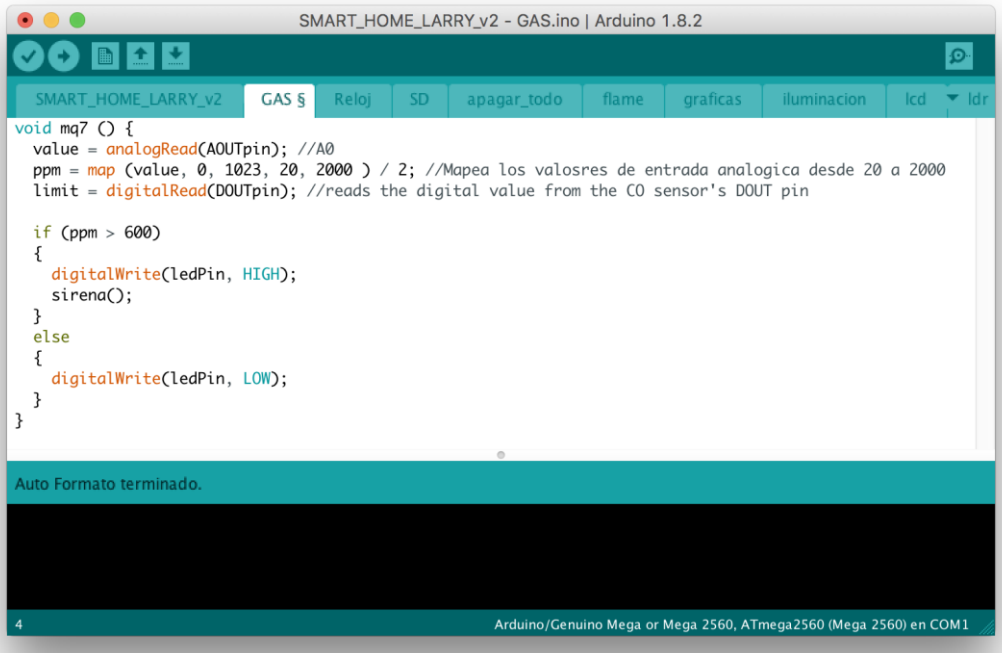

Imagen 26: Código sensor de gas

5.2.7. Iluminación

Con la función rele_iluminacion() comprobamos todos los valores del array, si ha habido cambios nuevos desde la app o la web, estos datos se guardan en la tarjeta SD y se actualiza en la salida digital los pines conectados a los relés.

5.2.8. Control Luminosidad

La función ldr() recoge información del sensor, si los datos son superiores a 900 lux la a otra función para que abra la persiana y en el caso que detecte por debajo de los 50 lux llama a otra función para que cierre la persiana

5.2.9. Puertas / Servos

Esta función puertas() modifica las salidas digitales posicionando el servo en 0 grados para cerrar la puerta y en 90 grados para abrir la puerta.

5.2.10. Detección de incendios

Con la función flame(), programamos la activación de un LED y un Buzzer cuando la señal del sensor nos de 0.

Imagen 27: Código sensor de llama

5.2.11. Sirena

Con la función sirena() activamos la salida PWM conectada al buzzer.

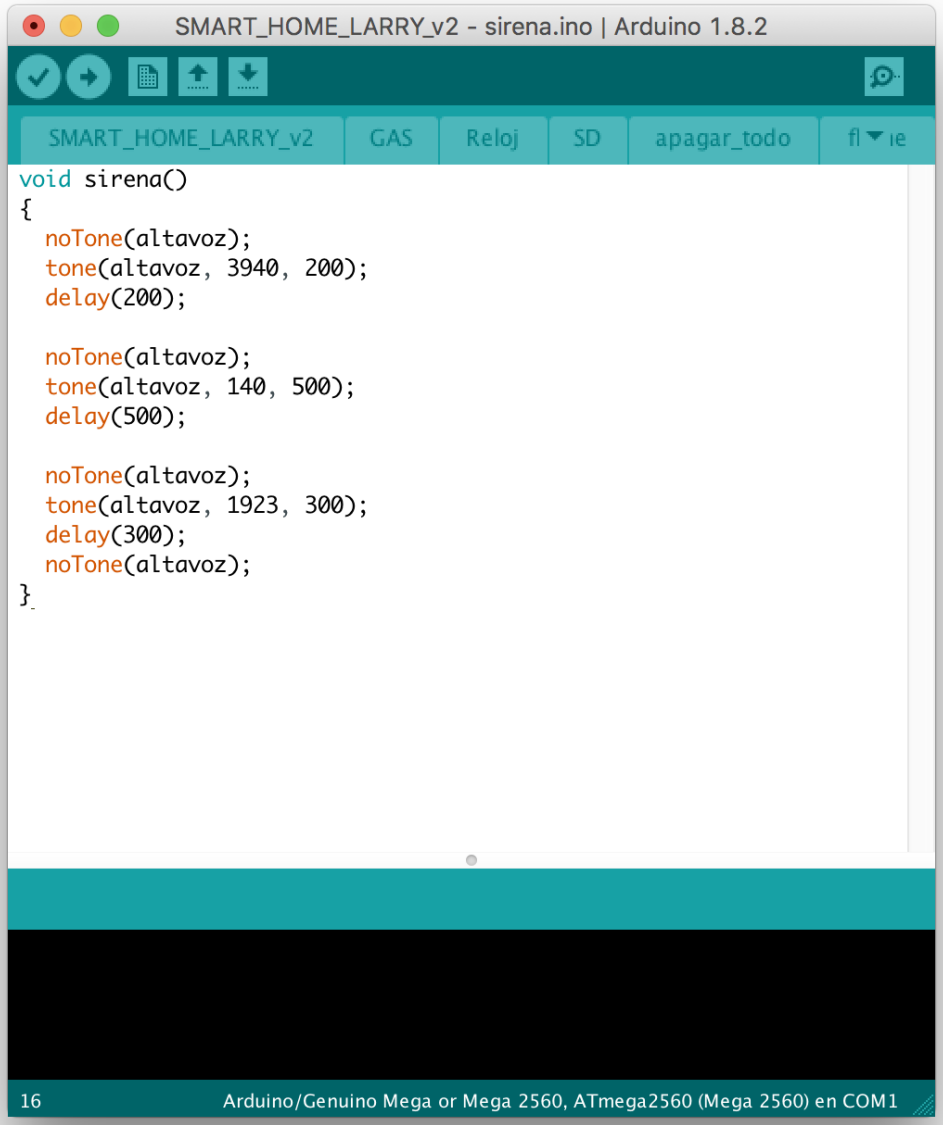

Imagen 28: Código activación de sirena

5.2.12. Apagar todas las luces

Con la función apagartodo() recorremos cada uno de los estados de los relés actualizando su nuevo valor a 0 y guardando a su vez este nuevo dato en la tarjeta SD.

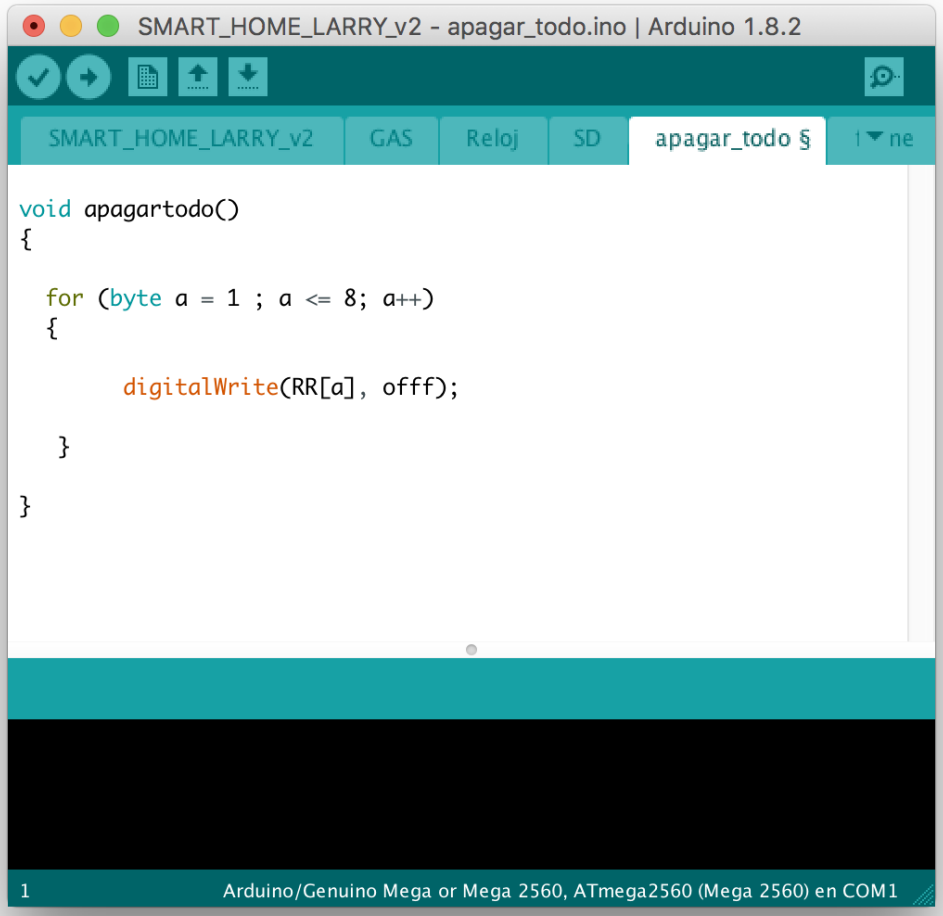

Imagen 29: Código apagar toda la iluminación

5.2.13. Base de datos

Recuperamos y guardamos los datos desde la tarjeta SD conectada a arduino por comunicación I2C.
guardar() abre el archivo *.txt o *.csv y guarda el nuevo dato.
guardarpermanente() guarda los datos más relevantes, usuario y contraseña.
INT_SD() abre el archivo *.txt o *.csv y extrae el dato y lo convierte a un valor entero tipo int.
txt_txt() recupera datos de la SD y los convierte a cadena de texto tipo String.

txt_int() recupera datos de la SD y los convierte a número tipo int.
num(fila, columna) obtiene un datos según la fila y columna y lo convierte a número tipo int.
filascsv() obtiene el número total de filas de un documento.
R_SD() Guarda los datos en la sd según el relé.
RR_SD() Recupera los datos en la SD de cada relé.
P_SD() Guarda los datos en la sd según la puerta/servo.
PP_SD() Recupera el dato de las puertas/servos desde la SD.

5.2.14. HTTP

La comunicación entre la app y arduino recibida es procesada con los siguiente funciones:

- StrClear() limpia la caché
- StrContains() obtiene el nombre de un argumento de la url
- tabla() obtiene el valor de un argumento de la url
- enviarsolicitud() envía el encabezado Http
- enviarsolicitudcss() envía el encabezado http para archivos css
- enviaracliente() envía una respuesta al cliente con sesión http activa
- pantalla() inicializa una nueva sesión si hay datos en la caché y de tiene la sesión después de contestar al cliente.
- XML_sensores() envía un xml con los valores de sensores en tiempo real.

5.2.15. RTC

La función reloj() obtiene los datos de hora, minuto, segundo, dia, mes año desde el chip DS1307.

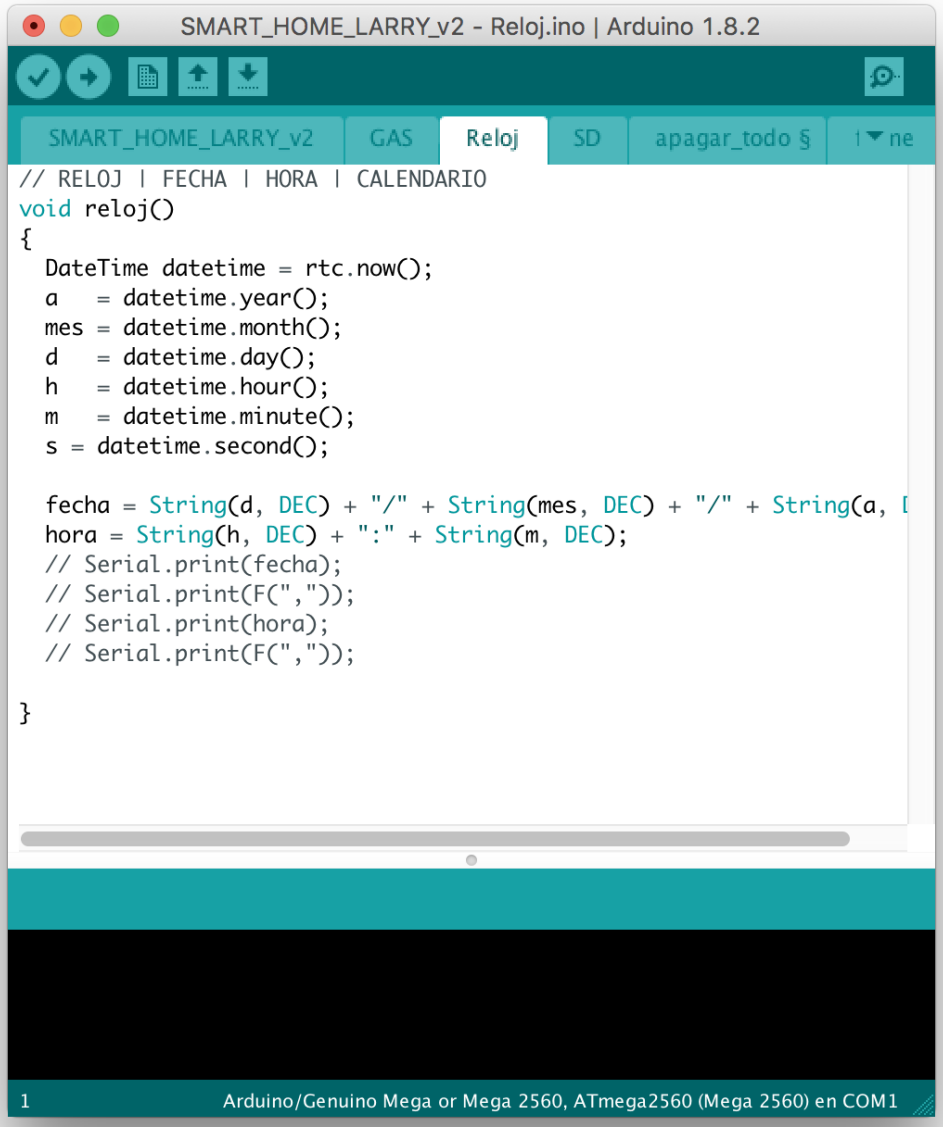

Imagen 30: Codigo del reloj

5.3. Interfaz Web

Para la implementación de la interfaz web he hecho dos apartados. En primer lugar, he abordado la parte estática la cual está desarrollada con HTML y CSS Responsive. Posteriormente he desarrollado la parte dinámica en Javascript.

6. Desarrollo

La maqueta tiene un tamaño 80cm x 60cm. Sobre esto se creará los ambientes necesarios para poner la placa Arduino y todos los sensores, actuadores y dispositivos que formen parte de la vivienda inteligente.

6.1. Proceso de construcción de la maqueta

La maqueta se va a construir de DM. Se utiliza un grosor de 5 milímetros. Las paredes tendrán una altura de 10 centímetros. Se pondrán unos tacos de madera para elevar la base y poder poner la mayoría del cableado por la parte inferior. Se ha realizado cortes y agujeros para pasar los cables y colocar los sensores.

Se coloca los puntos de iluminación, y se pasa el cableado correspondiente. Para la unión del cableado se usa soldadura de estaño y lo cubrimos con silicona.

Posteriormente colocamos los relés que nos permitirá mediante Arduino interactuar con cada punto de iluminación.

Colocamos todos los sensores, la pantalla LCD, motores, el zumbador y la placa Arduino.

Y por último pasamos todo el cableado que une cada sensor con Arduino, y cubrimos todas las soldaduras.

7. Viabilidad económica

En el siguiente apartado contiene un desglose de coste económico y trabajo.

7.1. Coste de inversión

- El coste de diseño se estima en unas 100 horas de trabajo.
- El coste de programación se estima en unas 400 horas.
- El coste de taller se estima 80 horas.

Componentes	Proveedor	Unidades	Precio Unitario	Precio Total
Maqueta	Carpintero	1	45	45
Placa Arduino 2560	Aliexpress	1	30	30
Shield Ethernet	Aliexpress	1	5	5
Cable Ethernet	Aliexpress	1	2	2
Tarjeta SD 4GB	Carrefour	1	6	6
Fuente Alimentación	Onda Radio	1	25	25
Cable 2mm	Onda Radio	4	2,5	10
Relé 8	Aliexpress	1	4	4
Cable 5mm	Onda Radio	6	0,7	4,2
Luces Led	Carrefour	8	2	16
Casquillos	China	8	0,2	1,6
Regletas	Onda Radio	4	1	4
Servos	Aliexpress	3	2	6
MQ135	Aliexpress	1	1,3	1,3
DTH11	Aliexpress	2	2	4
Flama	Aliexpress	1	2	2
LCD Crystal Liquid	Aliexpress	1	4	4
Buzzer	Aliexpress	1	0,1	0,1
Led rojo	Aliexpress	1	0,1	0,1
DS1307	Aliexpress	1	2	2
LDR	Aliexpress	1	1	1
Ventilador	Aliexpress	1	0,5	0,5
Pegamento madera	Carpintero	1	10	10
Router	Media Mark	1	25	25
Transistor NPN	Onda Radio	2	0,2	0,4
Resistencias	Onda Radio	-	-	3
Silicona	China	4	1	1
TOTAL				213,2

Tabla 2: Listado precio de componentes

7.2. Coste de utilización

Este dispositivo tiene un consumo de 20W/h y requiere un espacio para su instalación, medio metro cuadrado.
Preferiblemente se puede adaptar una zona de la casa que este acondicionada para ella.

7.3. Financiación

Este dispositivo se ha financiado con capital privado, aportando la cuantía por adelantado con intención de recuperar lo invertido. En una futura réplica y entrada al mercado de este mismo proyecto para uso de un tercero, se cobrará el total de lo invertido más un royalty de uso.

7.4. Análisis de viabilidad económica

Para introducir en el mercado este dispositivo no sería necesario hacer el diseño desde cero, ni la aplicación web.
Para un estudio detallado para una vivienda estándar se prevé el siguiente desglose.

- El coste de diseño se estima en unas 0 horas de trabajo.
- El coste de programación se estima en unas 0 horas.
- El coste de taller se estima 16 horas.
- El coste de componentes 213,2€.

La producción en serie de este dispositivo puede reducir hasta un 50% del coste de los componentes.

8. Cronograma

CRONOGRAMA		SEMANAS																					
FASES	ACTIVIDADES	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
ESTABLECIMIENTO DE OBJETIVOS	Proponer alternativa y valoración de las condiciones e hipótesis																						
	Analizar y redactar funcionalidades																						
EXPLORAR TECNOLOGIAS	Estudio de placa y microcontroladores																						
	Estudio de framework para aplicaciones web																						
FORMACION	Electrónica																						
	Programación C++, HTML, CSS, JavaScript																						
	Framework Ionic y Cordova																						
ADOPCION DE TECNOLOGIA	Explorar conceptos, integrarlos dentro del sistema para crear una beta y validar hipótesis.																						
CONVERGENCIA DE DISEÑO Y TECNOLOGIA	Diseño de conexiones eléctricos																						
	Programar microcontrolador																						
	Programar diseño de interfaz																						
	Determinar la experiencia de usuario																						
	Programar módulos de la interfaz																						
PROTOTIPADO	Organización de la implementación																						
	Pruebas piloto con la tecnología																						
	Test de posibles fallos																						
IMPLEMENTACION	Análisis del usuario con el prototipo																						
	Implementación de mejoras																						
	Test de calidad y depuración de fallos																						
	Mejoras de software																						
DOCUMENTACION	Redacción de documento final																						

Tabla 3: Cronograma

9. Conclusiones

Los objetivos alcanzados en la construcción del proyecto son los siguientes:

Hardware:

- Controlar las 7 luces LED para la iluminación.
- Controlar 3 servos para mover puertas y persiana.
- Alarma acústica (Buzzer).
- Alarma lumínica (Led Rojo).
- Ventilador para refrigeración de la casa.
- Reloj para tener la hora y fecha correcta en cada momento.
- Monitorizar sensores desde la web/app.
- Display 16x2 crystal liquid para monitorizar sensores sin entrar en la web/app.
- Detectar fuego, fuga de gas y notificar.
- Algoritmo control climático.
- Algoritmo control de luminosidad exterior.

Los objetivos no alcanzados:

Hardware:

- Teclado control de acceso. (keypad).
- Contador de consumo eléctrico de la casa.
- Cámaras IP.
- Baterías.
- Captadores de Energía Solar y Eólica.
- Bomba de riego para jardín.
- Control remoto IR para dispositivos multimedia.

Los objetivos alcanzados en desarrollo e implementación del proyecto son los siguientes:

Software:

- Lógica interface Web/App.
- Compilar interface para Web, Android, IOS, Windows.
- Capa de interface responsive.
- Base de datos.

- Acceso.
- Vista del plano de la casa con información en tiempo real.
- Notificaciones.

Los objetivos no implementados son los siguientes:
Software:

- Registro.
- Gestión de roles de usuario.
- Acceso parcial a las instalaciones de la Casa.
- Número de Usuario conectados.
- Historial de Notificaciones.
- Gráficas de temperatura, humedad, Gas, etc.
- Chat interno entre usuarios de la casa.
- Calendario.
- Añadir eventos según los datos de los sensores.
- Despertador.

9.1. Propuestas de mejoras

En este proyecto se ha cubierto sólo una parte de todas las funcionalidades que una casa inteligente puede tener hoy en día. Todos los objetivos no alcanzados se proponen como mejoras de la casa domótica.

10. Bibliografía

[1] ARDUINO. http://arduino.cc

[2] W3SCHOOLS. http://www.w3schools.com

[3] WIKIPEDIA. http://wikipedia.es

[4] NODEJS. https://nodejs.org/es

[5] Cordova. https://cordova.apache.org

[6] Ionic. https://ionicframework.com

[7] Brackets. https://brackets.io