



**Universidad
Europea Madrid**

LAUREATE INTERNATIONAL UNIVERSITIES

Informe Proyecto Integrador ArdVa

PROYECTO DE INGENIERÍA

CURSO 2016 - 2017

Miguel Ángel del Cerro Recuero

Rebeca Pérez Carrasco

Pablo Olivera Maganto

Carlos López Merino

Pablo Fernández Gonçalves

1. Introducción	2
2. Objetivos del proyecto	3
3. Diseño del sistema	4
4. Descripción de los elementos HW	6
Arduino Mega	6
LDR	7
Sensor DHT	7
Sensor de ultrasonidos HC-SR04	8
Sensor de corriente eléctrica no invasivo STC-013	9
Servo	10
Zumbador	11
RTC	12
Display LCD	12
Mando y receptor IR	13
Relés	13
5. Desarrollo SW	15
6.1 Arduino	15
6.2.1 Java Interfaces	16
6.2.2 Java BBDD	17
6. Pruebas	19
7. Trabajos Futuros y Líneas de Mejora	20
8. Conclusiones	21

1. Introducción

El presente proyecto trata de la gestión de domótica del hogar con Arduino y Java, realizado por un grupo de alumnos de la asignatura de Proyecto de Ingeniería, muestra la integración de un sistema de domótica mediante Arduino y Java, indicamos los funcionamientos que se realizan en las diferentes plataformas.

Es un hecho que la domótica está generando gran impacto en la sociedad, y que en un futuro no muy lejano, gracias a las ventajas que ofrece. Y no solo la domótica se está posicionando fuerte, en nuestra sociedad hay un boom tecnológico por así decirlo, en cuanto al uso de móviles inteligentes y tabletas.

La aplicación tiene como fin combinar el uso de dispositivos conectado a Arduino con la domótica, creando un prototipo de sistema de control domótico, el cual permitirá controlar, relé, alarmas, zumbador, sensor ultrasonidos, entre otros.

A continuación, describiremos la estructura que tiene la memoria y resumimos el contenido de cada capítulo analizando cada uno de pasos llevados a cabo por el grupo.

2. Objetivos del proyecto

Es un proyecto que implica la creación de un sistema de gestión domótica del hogar, que integre Java y Arduino.

Por un lado, se implementó de forma que Arduino fuera capaz de llevar a cabo una serie de funciones de forma autónoma (control de climatización, control de iluminación y mediciones de consumo). Así mismo se diseñó para que cada cierto tiempo Arduino proveyera de datos de los sensores a Java para su incorporación en una base de datos.

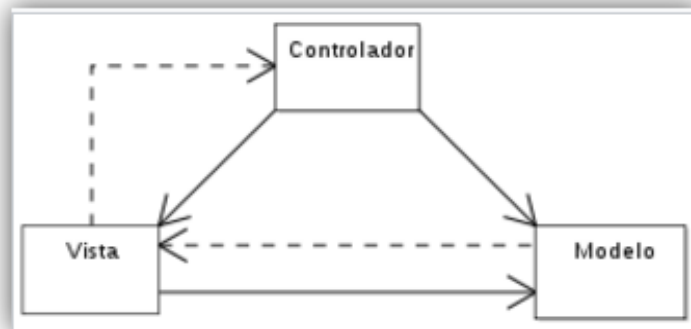
Por otro lado, se diseñó una interface para Java que permite: tanto monitorizar los sensores de Arduino mientras funciona de forma autónoma, como actuar de forma manual sobre los relés que controla Arduino. También se decidió incluir una función de alarma que, si bien Arduino es capaz de llevar a cabo de forma autónoma, necesita de la indicación de Java para activar o desactivar esa función.

También se decidió poner una ventana que muestre los valores que se han almacenado en la base de datos.

3. Diseño del sistema

a. Pasarela

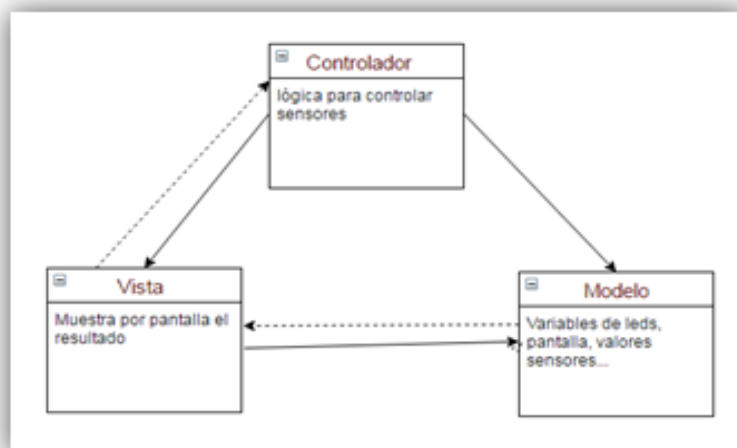
Nuestro desarrollo seguirá el diseño de Modelo-Vista-Controlador (MVC) con el fin de separar los datos y la lógica de negocio de nuestro sistema de la propia interfaz de usuario y el módulo que se encargará de gestionar los eventos y las comunicaciones, en este caso, de gestionar los eventos de los sensores y su comunicación.



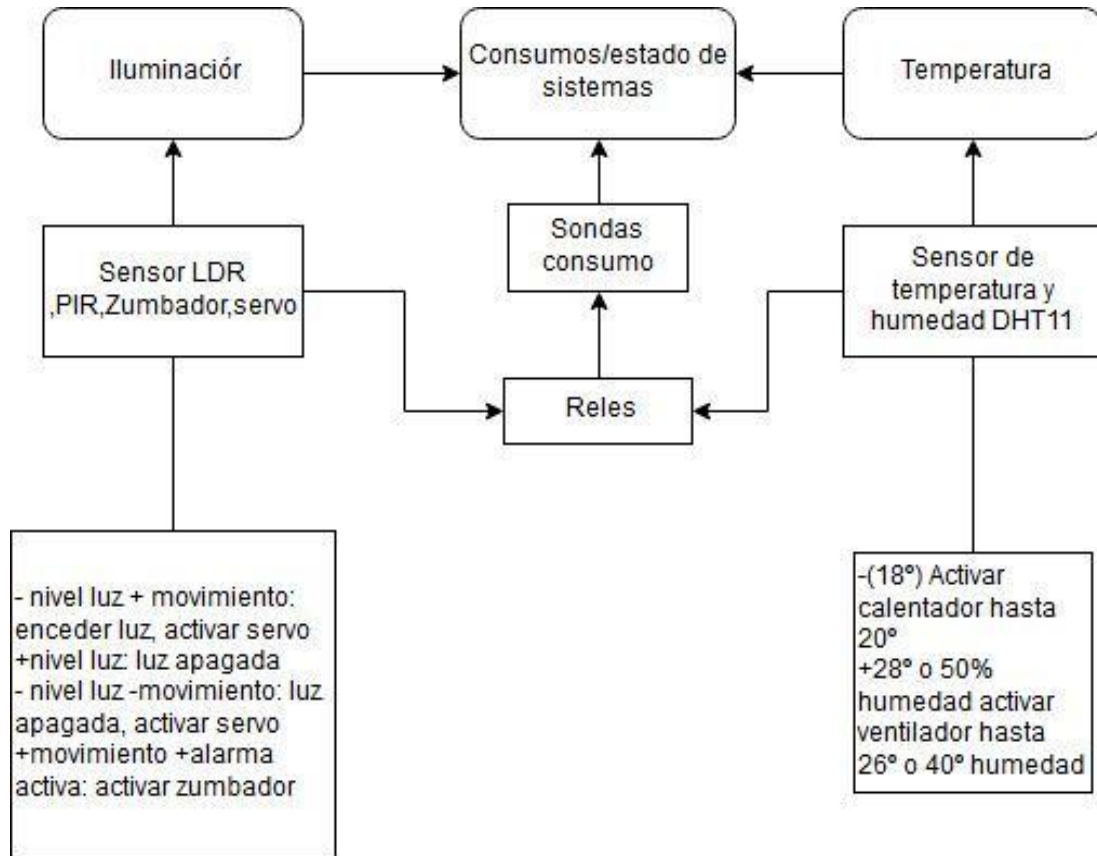
De tal forma que nuestro diseño de termostato en un diagrama de clases siguiendo este modelo quedaría de la siguiente manera:

- **Modelo:** En él estarán el conjunto de todas las variables del diseño: leds, pantalla, valor de los sensores.
- **Controlador:** En él meteremos la lógica para controlar los sensores.
- **Vista:** Mostraremos por pantalla todo el resultado el resultado de la interacción del usuario con nuestro desarrollo.

De tal forma que el diagrama de clases resultaría como sigue a continuación:



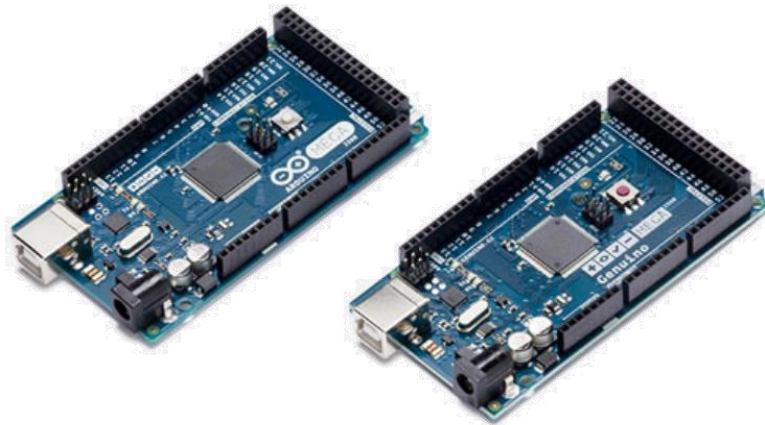
b. Esquema de funcionalidades



4. Descripción de los elementos HW

Arduino Mega

En nuestro proyecto vamos a integrar todo el montaje usando un modelo Arduino Mega 2560; ya que tiene algunas características que lo hacen más ventajoso para nuestro proyecto, mientras que las diferencias como el sobrecoste o el aumento de tamaño son poco relevantes en este caso concreto.



Entre las principales ventajas de este modelo respecto a otros de la familia está la mayor cantidad de entradas/salidas disponibles (54 entradas/salidas digitales, 15 de ellas con pwm; 16 entradas analógicas), y la mayor cantidad de memoria disponible (256 KB de memoria flash y 8 KB de memoria SRAM).

Entre las características compartidas con otros modelos Arduino están el modo de alimentación: teniendo un voltaje de funcionamiento de 7-12V en caso de usar su conector de alimentación estándar, o 5V en caso de alimentar directamente al Arduino a través de los pines Vcc/GND. Así mismo tiene la capacidad de establecer conexión con otros dispositivos mediante comunicaciones serie, SPI, TWI, I2C, One Wire (en algunos casos puede necesitar librerías adicionales y en otros están entre las incluidas en el IDE).

También hay que destacar que la mayoría de las shield utilizadas en el modelo UNO son compatibles con el modelo MEGA, cuidado con las que utilicen la comunicación I2C ya que se encuentran en diferentes ubicaciones y probablemente no sean compatibles.

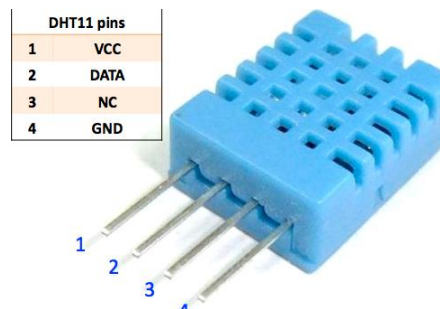
LDR



Las resistencias LDR cambian su valor de resistencia en función de la luz que reciban, según el modelo a utilizar el valor de su resistencia es diferente, pero todas actúan de la misma manera.

Para su uso es necesaria una entrada analógica para realizar las medidas del sensor.

Sensor DHT



El DHT11 y el DHT22 son dos modelos de una misma familia de sensores, que permiten realizar la **medición simultánea de temperatura y humedad**.

Estos sensores disponen de un procesador interno que realiza el proceso de medición, proporcionando la medición mediante una señal digital, por lo que **resulta muy sencillo obtener la medición desde un microprocesador como Arduino**.

Ambos sensores presentan un encapsulado de plástico similar. Podemos **distinguir ambos modelos por el color del mismo**. El DHT11 presenta una carcasa azul, mientras que en el caso del sensor DHT22 el exterior es blanco.

De ambos modelos, el DHT11 es el hermano pequeño de la familia, y cuenta peores características técnicas. El DHT22 es el modelo superior, pero por contra, tiene un precio más elevado.

Las características del DHT11 son realmente escasas, especialmente en rango de medición y precisión.

- Medición de temperatura entre 0 a 50, con una precisión de 2°C
- Medición de humedad entre 20 a 80%, con precisión del 5%.
- Frecuencia de muestreo de 1 muestras por segundo (1 Hz)

El DHT11 es un sensor muy limitado que podemos usar con fines de formación, pruebas, o en proyectos que realmente no requieran una medición precisa.

Por el contrario, el modelo DHT22 tiene unas características mucho más aceptables.

- Medición de temperatura entre -40 a 125, con una precisión de 0.5°C
- Medición de humedad entre 0 a 100%, con precisión del 2-5%.
- Frecuencia de muestreo de 2 muestras por segundo (0.5 Hz)

EL DHT22 (sin llegar a ser en absoluto un sensor de alta precisión) **tiene unas características aceptables** para que sea posible emplearlo en proyectos reales de monitorización o registro, que requieran una precisión media.

En nuestro diseño, por una cuestión de disponibilidad de componentes, se ha optado por utilizar el sensor DHT11, aunque sería fácilmente adaptable al DHT22.

Sensor de ultrasonidos HC-SR04



Es un dispositivo medidor de distancias. Está basado en el envío de un pulso de alta frecuencia, que rebota cuando golpea con un obstáculo y es reflejado al sensor que dispone de un micrófono adecuado para la recepción de esa frecuencia. Midiendo el tiempo transcurrido entre el envío del pulso y la recepción del mismo, y conociendo la velocidad del sonido, se puede calcular la distancia hasta el obstáculo.

El rango teórico de medición de distancias está en unos 2-400 cm, no obstante, en la práctica su alcance suele quedar limitado a unos dos metros. Tienen una precisión bastante

limitada, lo que ocasiona que, en caso de haber un gran número de objetos cercanos o con superficies irregulares, la medición puede falsearse. No obstante, en nuestro caso lo utilizamos a modo de fotocélula, pudiendo emplearse para ver si alguien ha entrado en una habitación.

Sensor de corriente eléctrica no invasivo STC-013



La familia STC-013 son sensores de corrientes no invasivos que permiten medir la intensidad que atraviesa un conductor sin necesidad de cortar o modificar el conductor. Podemos emplear estos sensores con un procesador como Arduino para medir la intensidad o potencia consumida por una carga

Los sensores STC-013 son transformadores de corriente, dispositivos de instrumentación que proporcionan una medición proporcional a la intensidad que atraviesa un circuito. La medición se realiza por inducción electromagnética.

Los sensores STC-013 disponen de un núcleo ferromagnético partido (como una pinza) que permite abrirlo para arrollar un conductor de una instalación eléctrica sin necesidad de cortarlo.

Dentro de la familia STC-013 existen modelos que proporcionan la medición como una salida de intensidad o de tensión. Dentro de lo posible, lo normal es que prefiramos salida por tensión porque la conexión es más sencilla.

La precisión del sensor puede ser de 1-2%, pero para ello es muy importante que el núcleo ferromagnético se cierre adecuadamente. Hasta un pequeño hueco de aire puede introducir desviaciones del 10%.

Como desventaja, al ser una carga inductiva, el STC-013 introduce una variación del ángulo de fase cuyo valor es función de la carga que lo atraviesa, pudiendo llegar a ser de hasta 3°.

En nuestro proyecto usamos unas sondas con capacidad 30A/1V, que son las más próximas a valores de consumo utilizados en una casa. No obstante, para medidas de bajo consumo su precisión no es tan óptima y sería necesario optar por otro modelo de sondas.

Servo



Un servo de rotación continua es una variante de los servos normales, en los que la señal que enviamos al servo controla la velocidad de giro, en lugar de la posición angular como ocurre en los servos convencionales.

Otra diferencia con los servos convencionales, que tienen un rango limitado de movimiento de 0 a 180°, es que un servo de rotación continua puede girar 360 grados en ambos sentidos de forma continua.

Las características y el control de un servo de rotación continua son similares a los de un servo convencional, de hecho, es posible modificar un servo para convertirlo en rotación continua simplemente eliminando los topes internos y sustituyendo el potenciómetro interno por dos resistencias iguales. No obstante, el rendimiento será inferior a un servo de rotación continua comercial.

Muchos modelos de servo de rotación continua incluyen un potenciómetro de calibración que permite ajustar con precisión el punto neutro, es decir, el punto en el que el servo no gira en ninguno de los sentidos (siendo otra ventaja de usar un motor comercial en lugar de un servo convencional modificado).

Los servos de rotación continua, al igual que los servos convencionales, admiten tensiones de alimentación de entre 4,8V a 7,2V. También al igual que sus hermanos incorporan un reductor interno por lo que, en general, proporcionan un alto par y baja velocidad máxima, en torno a 1-2 rpm.

Zumbador



Los buzzer activos, en ocasiones denominados zumbadores, son dispositivos que **generan un sonido de una frecuencia determinada y fija** cuando son conectados a tensión.

Un buzzer activo incorpora un oscilador simple por lo que **únicamente es necesario suministrar corriente al dispositivo** para que emita sonido. En oposición, los buzzer pasivos necesitan recibir una onda de la frecuencia.

Al incorporar de forma interna la electrónica necesaria para hacer vibrar el altavoz **buzzer activo resulta muy sencillo de conectar y controlar**. Además, no suponen carga para el procesador ya que no este no tiene que generar la onda eléctrica que se convertirá en sonido.

En contraposición, tienen la desventaja de que **no podremos variar el tono del sonido emitido**, por lo que no podremos realizar melodías, cosa que sí podremos hacer con los buzzer pasivos.

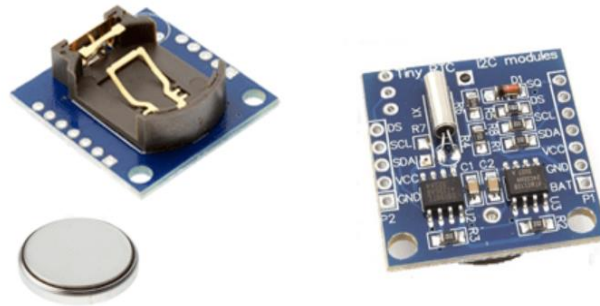
Físicamente pueden ser muy parecidos, o incluso idénticos, a los buzzer pasivos, por lo que puede llegar a ser difícil determinar a simple vista si un buzzer es activo o pasivo.

Existen buzzer activos en un gran abanico de tamaños y potencias, desde tonos casi imperceptibles hasta alarmas realmente estridentes. El consumo eléctrico, lógicamente, también varía con la potencia del buzzer.

Podemos emplear los buzzer activos de menor potencia, por ejemplo, para dar avisos al usuario o proporcionar un feedback ante alguna acción, como pulsar un botón, para que el usuario compruebe que su acción ha sido recibida.

Por su parte, los buzzer de mayor potencia son adecuados para generar alarmas de forma sencilla, por ejemplo, combinados con un sensor de movimiento, un sensor de agua, o un sensor de llama, entre otros.

RTC



Debido a que Arduino no tiene forma de guardar la fecha y hora utilizaremos un RTC DS1307 para mantener la hora, y que pueda realizar funciones específicas en función de la hora y mostrar en pantalla los valores. Necesitan el uso de una pila CR2032 para poder almacenar la hora durante los periodos de tiempo en que no esté alimentado externamente.

Este módulo requiere el conexionado a Arduino mediante bus I2C, por lo que necesita cuatro pines para su funcionamiento: Vcc, GND, SDA, SCL.

Display LCD



Las pantallas LCD son una de las formas más fáciles y económicas de proporcionarle un display a un autómeta como Arduino. Están disponibles en diferentes tamaños siendo 16x02 (2 líneas de 16 caracteres) y 20x04 los modelos más comunes de encontrar. Algunos disponen de retroiluminación trasera en azul o verde, que puede ajustarse por medio de un potenciómetro.

En caso de utilizar un LCD convencional serían necesarios 10 pines (3 de alimentación y 7 de datos) para poder funcionar, lo que demanda una gran cantidad de pines de nuestro Arduino. De modo que se ha optado por un modelo que lleva adosado un controlador LCD I2C, el cual permite controlar el LCD con solo 4 pines (2 de alimentación y 2 de datos).

Mando y receptor IR



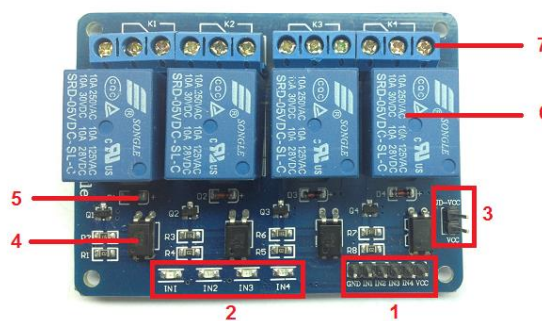
El mando a distancia emplea un LED infrarrojo para enviar una señal deseada al receptor. Al tratarse de un LED infrarrojo y no de radiofrecuencia hay que recordar que necesita línea directa entre receptor y mando para poder funcionar

Un mando a distancia emplea un emisor de luz en el infrarrojo cercano, invisible para el ojo humano, pero que puede ser captado con facilidad por un receptor infrarrojo. Es posible ver la luz del mando mirando el LED infrarrojo con una cámara digital, por ejemplo, de un móvil.

El alcance es limitado, típicamente inferior a 3m. La distancia depende fuertemente del ángulo de emisión, disminuyendo rápidamente a medida que nos desviamos de la dirección frontal.

También existen emisores LED infrarrojos que harían posible emplear Arduino para, primero clonar un mando a distancia, y después controlar el encendido de una televisión, un equipo de música o el aire acondicionado.

Relés



Como se puede apreciar, la placa tiene un conector de entradas (IN1 a IN4) y alimentación (GND es masa o negativo y Vcc es el positivo) [1], cuatro leds que indican el estado de la entradas [2], un jumper selector para la alimentación de los relés [3], cuatro opto acopladores del tipo FL817C [4], cuatro diodos de protección [5], cuatro relés marca SONGLE con bobinas de 5V y contactos capaces de controlar hasta 10 Amperios en una tensión de 250V [6] y cuatro borneras, con tres contactos cada una (Común, Normal abierto y Normal cerrado), para las salidas de los relés [7].

Hay que destacar que los relés se activan en estado bajo, con lo cual para activarlos se necesita enviar desde Arduino “LOW” (0V) y para desactivarlos “HIGH” (5V).

5. Desarrollo SW

6.1 Arduino

Para la programación de Arduino se ha usado el IDE 1.8.2.

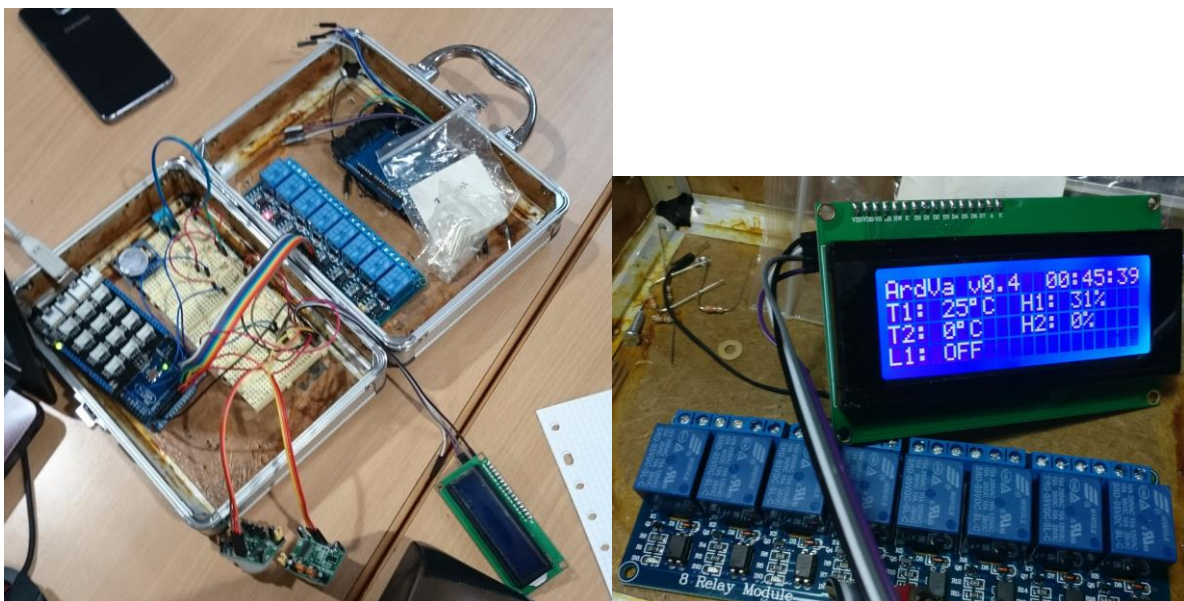
Se ha hecho uso de algunas librerías propias del entorno de programación de Arduino:

- Servo.h → Permite un fácil manejo de servos desde Arduino
- Wire.h → Permite la comunicación con dispositivos I2C

Así mismo se han utilizado librerías de terceros:

- DHT.h → Usada para el manejo de la familia de sensores DHT (11, 21,22)
- EmonLib.h → Usada para las medidas realizadas con las sondas de consumo
- IRremote.h → Para trabajar con comunicaciones con un receptor/emisor IR
- LiquidCrystal_I2C.h → Manejo display LCD alfanuméricos protocolo I2C
- RTCLib.h → Simplifica el uso de un RTC DS1307

Se diseñó un protocolo de comunicaciones entre Arduino y Java: cuando Java quería que Arduino simplemente enviaba un código de 4 dígitos (acabado en cero o uno) con la petición codificada; cuando Java quería saber el estado de “algo” en Arduino, se enviaba igualmente un código de 4 dígitos (acabado en nueve) y Arduino respondía con él mismo código seguido del valor resultante.



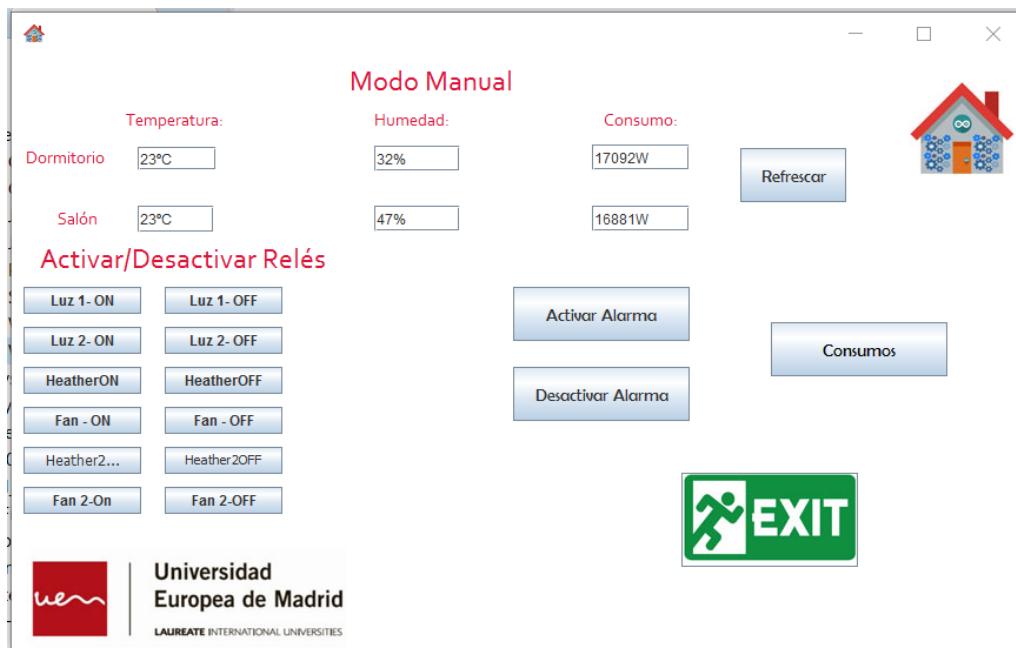
6.2.1 Java Interfaces

Se ha realizado la codificación de la pasarela con el nodo central, y la conexión con las interfaces gráficas. Usando la librería Swing de java para el diseño de las interfaces, centrándonos en tres paneles diferenciados.

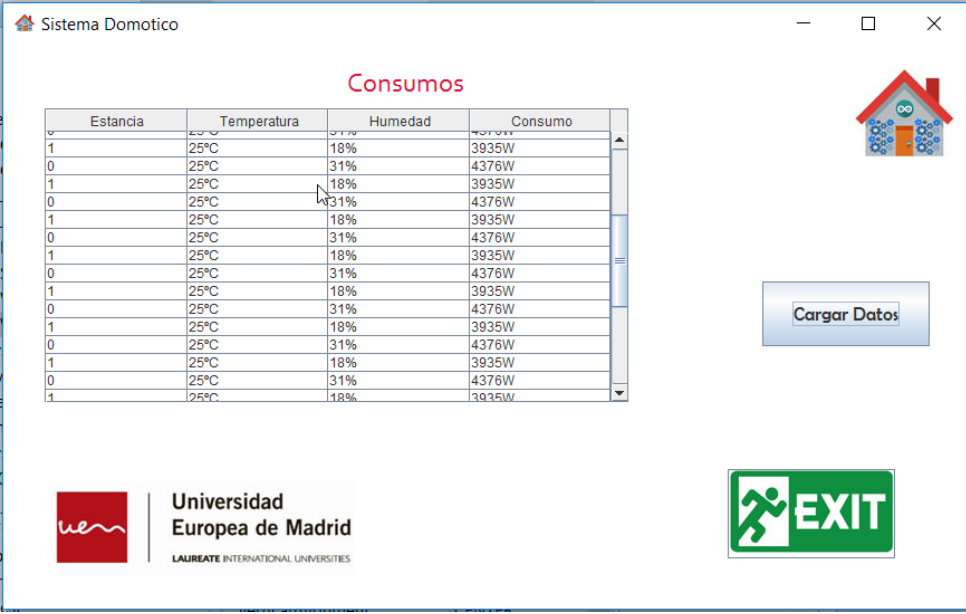
Panel principal:



Panel manual:



Panel consumos:



Consumos

Estancia	Temperatura	Humedad	Consumo
1	25°C	18%	3935W
0	25°C	31%	4376W
1	25°C	18%	3935W
0	25°C	31%	4376W
1	25°C	18%	3935W
0	25°C	31%	4376W
1	25°C	18%	3935W
0	25°C	31%	4376W
1	25°C	18%	3935W
0	25°C	31%	4376W
1	25°C	18%	3935W
0	25°C	31%	4376W
1	25°C	18%	3935W
0	25°C	31%	4376W
1	25°C	18%	3935W

Cargar Datos

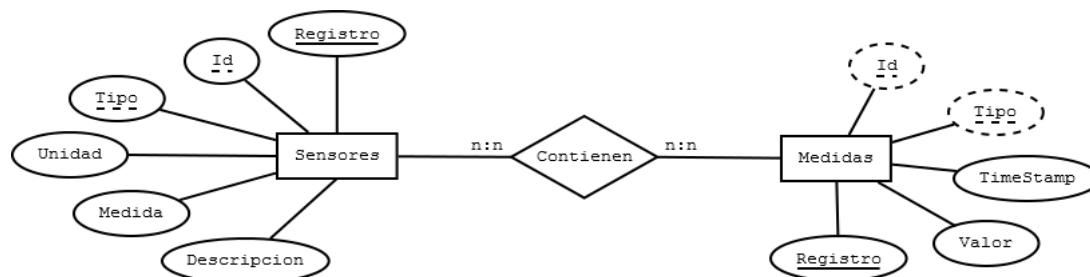
Universidad Europea de Madrid
LAUREATE INTERNATIONAL UNIVERSITIES

EXIT

6.2.2 Java BBDD

Desde que decidimos implementar una Base de Datos a nuestro proyecto, hemos pasado por varios diseños ER y estructuras intentando conseguir la mayor eficacia a la hora de recoger los datos, disponer de ellos y manejarlo. Además, pensando en el futuro, hemos diseñado la BBDD de manera que, si queremos añadirle más funcionalidades, sensores o dispositivos finales, no tengamos más que realizar un pequeño cambio en la BBDD añadiendo en las tablas los nuevos campos.

- Entorno PhPMyAdmin.y lenguaje SQL
- Interacción con Java para recogida y consulta de datos
- Sistema modular con vista al futuro gracias al diseño y la programación.



	Tipo	Id
Sensor LDR:	1	0-1
DHT(Temperatura)	2	0-1
DHT(Humedad)	3	0-1
PIR	4	0-1
Sonda	5	0-2

6.2.3 Pasarela

Se han unificado las funciones en un solo programa, con lo que se permiten las siguientes funciones:

- Recibe señales de control por puerto serie: válidas tanto para solicitar información de los sensores, como para cambiar el modo de funcionamiento (automático/manual), como para interactuar con los relés.
- Por otro lado, tiene varias funciones autónomas que permiten: el control de la iluminación (mediante valores obtenidos con un sensor de movimiento y una fotorresistencia), el control de la climatización (mediante valores obtenidos de un sensor DHT11)
- También se han definido las funciones para que el sistema sea fácilmente escalable, ya que al llamar a las funciones de control hay que indicar el sensor a leer y el relé sobre el que actúa. De modo que en caso de querer añadir nuevas habitaciones a controlar solo es necesario indicar el nuevo sensor y actuador a utilizar, necesitando un cambio mínimo en el programa.
- Así mismo se ha realizado una función para un envío automático del estado de los sensores al Nodo cada 30 segundos. En realidad, el intervalo de envío es aproximado (28-32 segundos) ya que, al intentar hacer un envío cada un intervalo exacto de tiempos ocasionaba que algunos envíos de información se perdieran al estar ocupado el Arduino en otras tareas. Con la solución implementada se logra que no se pierda ningún envío de datos.
- Además, se ha añadido un display que permite ver desde la pasarela el valor de ciertos sensores, así como del estado de funcionamiento de algunos relés, lo que permite visualizar el funcionamiento sin necesidad de contar con el Nodo.

6. Pruebas

Se han ido realizando numerosas pruebas a lo largo del proyecto: tanto a título individual cuando cada uno iba realizando sus propias partes de código, como posteriormente en grupo cada vez que se iban implementando las nuevas funciones en cada nueva actualización del sistema.

Estas pruebas han hecho que hubiera que descartar algunos planteamientos iniciales:

- Por un lado, hubo que modificar el código inicial de Arduino donde usaba variables float, lo que le suponía una importante carga de trabajo a Arduino. Estos han sido sustituidos en la versión final con enteros, ya que no afecta al funcionamiento real y ralentizan menos el funcionamiento del sistema.
- Respecto al Display LCD, tardamos bastante tiempo en encontrar una librería que funcionara con el I2C y la nueva versión del Arduino.
- También hubo que prescindir de los sensores PIR, ya que tienen mucho retardo para empezar a funcionar y al unir las mediciones de los mismos con las mediciones de la resistencia LDR hacía imposible su funcionamiento; debido a lo cual hubo que acabar recurriendo al sensor de ultrasonidos.
- Dentro del apartado del interfaz, se han producido diversos cambios debido a las pruebas realizadas, como añadir botones y Labels para mostrar datos.
- En el apartado de la BBDD, hemos pasado por múltiples diseños hasta llegar a uno modular que aceptara cambios sin mucho esfuerzo en un futuro.
- Respecto a la puesta en común de todos los módulos, ha resultado trabajoso ya que, aunque todos los miembros conocemos el trabajo de los compañeros y hemos colaborado, cada uno se ha encargado de varios módulos específicos de manera individual por lo que, a la hora de hacer la puesta en común ha resultado costoso.
- Bloqueo de aplicación en la recogida de datos e inserción en la BBDD.

7. Trabajos Futuros y Líneas de Mejora

Algunos puntos que no pudieron implementarse en su momento y que serían muy interesantes de cara al futuro serían:

- Pon un lado, colocar la parte del Nodo en un ordenador que dispusiera de un servidor web. De modo que se pudiera interactuar sobre Arduino, no solo a través de cable (directamente en el nodo) o a través de WiFi (mediante tablet o smartphone, sino que se pudiera interactuar a través de internet estando en cualquier punto con una conexión de datos.
- También poder visualizar los valores almacenados en la base de datos en forma de gráficos sería un importante añadido. Ya que monitorizar cambios a largo plazo mirando tablas de valores resulta tedioso, mientras que visualizar las gráficas simplifica la operación.
- Montar el sistema domótico en una maqueta, le añadiría un valor más visual.

8. Conclusiones

Este proyecto ha servido para unir las capacidades de nuestro grupo, de manera que: por un lado, cada uno “lideraba” el ámbito en el que estaba más capacitado e iba solicitando lo que necesitaba al resto, y después de forma grupal todo se iba exponiendo y uniendo en conjunto y se iban haciendo las modificaciones necesarias para la siguiente revisión del proyecto.

Así mismo nos ha sido útil para unir conocimientos de distintas asignaturas y ver hasta donde se puede llegar cuando se integran.

Puede ser muy útil para cualquier persona que desee monitorear su casa, este es un claro ejemplo de un proyecto de domótica que se puede implementar en el hogar o en cualquier institución o infraestructura, con la ayuda de java se puede establecer una interfaz más cómoda e interactiva con el usuario. Demostrando que la tecnología puede ayudar a que la vida cotidiana sea más cómoda y con beneficios muy positivos.