PROYECTO NUESTRA TABLA PERIÓDICA

escuela técnica superior de ingeniería informática

UNIVERSIDAD DE SEVILLA

I.E.S. Virgen de Gracia
Oliva de la Frontera, BADAJOZ

Tutor del proyecto:

Luis Miguel Soria Morillo.

Integrantes del proyecto:

Jaime Cortés Vázquez.

Francisco Javier Viera Chaves.

Asier Herrería Oña.

PROYECTO NUESTRA TABLA PERIÓDICA

Introducción:

Antes de ahondar en componentes y funciones del proyecto propuesto hay que entender los objetivos impuestos por el plan docente de la asignatura “Desarrollo de Aplicaciones Distribuidas” dentro de la titulación Ingeniería de Computadores.

Como bien indica el nombre de dicha asignatura el proyecto tiene que estar basado en un sistema distribuido que haga uso de los diferentes recursos y tecnologías propuestos por el profesorado.

¿QUE ES UN SISTEMA DISTRIBUIDO?

Un sistema distribuido está formado por un conjunto de computadores autónomos entre si conectados por una red equipados con un sistema distribuido.

Podemos definir entonces un sistema distribuido como aquel en el que los componentes hardware o software, que se encuentran en los equipos enlazados por una red se comunican y coordinan mediante paso de mensajes.

Ejemplo de sistema distribuido.

Para conseguir esto se han propuesto dos placas: Arduino y ESP8266.

Estas son plataformas de software libre usados para modelar prototipos que utilizan un software y hardware con poca complejidad y fácil de usar (como es nuestro caso).

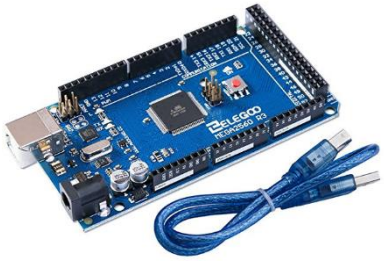
Nuestro objetivo será conectar Arduino a un servicio rest mediante la ESP8266, que de serie dispone de un módulo Wifi, e interactuar con él

mediante el paso de mensajes generados por los sensores conectados a la placa (en nuestro caso serán botones).

• Componentes:

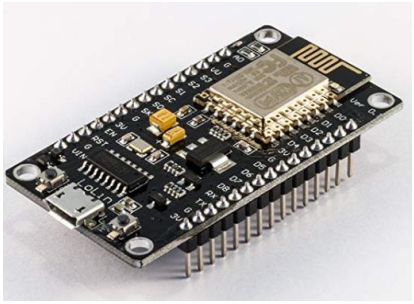
Para conseguir esto necesitaremos los siguientes componentes:

1. Placa Arduino.



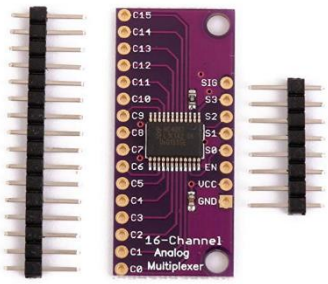
En esta placa irán todas las entradas y salidas utilizadas para conectar cada una de las celdas de la tabla periódica y poder comunicarnos así con el ESP8266.

2. Placa ESP8266NodeMCU.



Esta placa será la que recoja la información trasmitida por el Arduino y se encargue de comunicarse con el servidor web. Su funcionamiento es similar al de Arduino, el motivo por el cual la usamos es por el módulo Wifi que incorpora de serie y facilita de esta forma la comunicación con el servidor web.

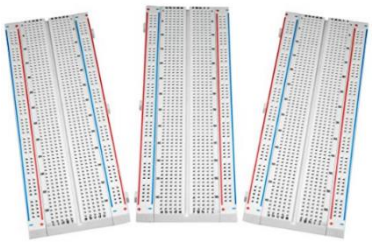
3. 9 Multiplexores CD74HC4067 de 16 canales.



Este circuito integrado nos permitirá multiplicar por 16 la capacidad de puertas de nuestro Arduino. Posee 16 canales en donde se encontrarán los botones correspondientes a cada uno de los elementos de la tabla periódica, una salida por donde recogeremos la información correspondiente al botón que se ha pulsado,

una señal de enable que permite el funcionamiento del circuito o lo inhibe y 4 entradas de selección que nos permitirán escoger cual de los canales (botones) ha sido activado y pasar a la salida un resultado binario de 4 bits en este caso.
Como son 118 elementos serían necesarios 9 multiplexores, cada uno de ellos ocupa 5 pines en el Arduino, 45 pines en total, por ello vamos a hacer uso de la placa Arduino grande de 52 pines y no la pequeña de 14 que es más barata pero no cumple con los requisitos.

4. 2 Packs de 3 protoboas.

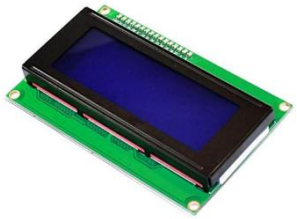


Estas placas nos proporcionaran las conexiones electrónicas necesarias entre los multiplexores, el Arduino y el ESP8266, sin necesidad de soldar.

5. Cables y botones.

Cables y botones necesarios para conectar cada uno de los elementos correspondientes a la tabla periódica con el circuito principal.

6. Actuadores (Pantalla LCD).



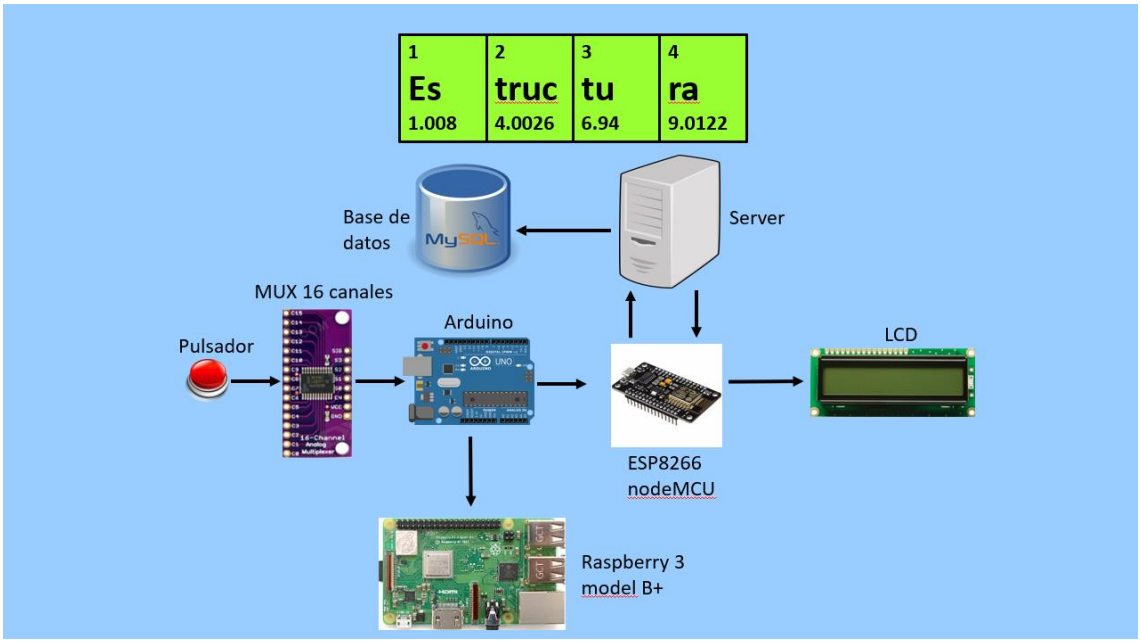
El LCD simplemente se encargará de informar que elemento es el que ha sido seleccionado y mostrará su nombre. ¿Por qué usamos esto si no es necesario para la realización del proyecto? Nos exigen hacer uso de unos actuadores que reaccionen a las comunicaciones que se realicen entre los componentes que forman la tabla periódica y el servidor rest, por lo que la forma más fácil y barata de implementar esto es añadiendo una pantalla LCD a la tabla periódica que se accione cuando haya comunicación entre nuestro sistema y el servidor.

Por último matizar que como es una aplicación distribuida necesitamos desarrollar una aplicación que interactué con nuestro sistema, por lo que hemos optado por desarrollar una aplicación para dispositivos Android que muestre los videos explicativos correspondientes a cada elemento que se guardaran en una base de datos a la que accederá el servidor web, y nos dará mucho juego si en un futuro se quieren implementar muchas más funcionalidades que por razones de tiempo en primera instancia no van a poder implementarse.

Somos conscientes de que esta prohibido el uso de teléfonos móviles en centros escolares por lo que indagando un poco hemos encontrado que haciendo uso de un miniPC (Rasberry Pi 3) que conste de sistema operativo Android e integrándolo en nuestro sistema, podría conectarse a cualquier televisor del centro que disponga de HDMI y cumplir con la función deseada sin necesidad de hacer uso de un smartphone o similar.

PROYECTO NUESTRA TABLA PERIÓDICA

- Esquema de conexiones y funcionamiento:



Como podemos ver los botones asociados a cada uno de los elementos de la tabla periódica irán conectados a los diferentes canales de los multiplexores, cuando se active un botón se generará una salida numérica que se identificara con el elemento al que corresponda el botón pulsado.

La salida obtenida tendrá que ser decodificada e interpretada por lo que ese número corresponderá con el numero asociado a cada elemento y de esta forma podremos saber cuál estamos escogiendo.

Ejemplo:

0 -> Hidrogeno (H)

10 -> Neón (Ne)

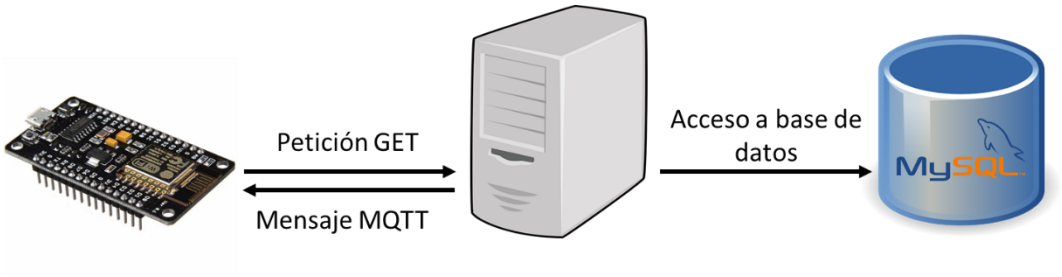
15 -> Fósforo (P)

Este mapeado se realizará con cada uno de los elementos de la tabla periódica.

PROYECTO NUESTRA TABLA PERIÓDICA

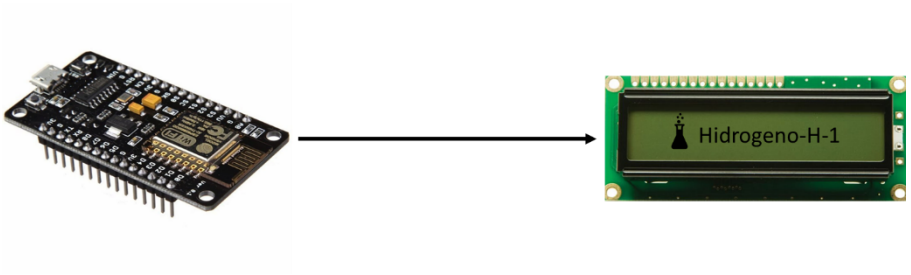
Para poder comunicar el Arduino con el ESP8266 ha sido necesario un puerto serie software, programado mediante la librería SoftwareSerial.h, ya que, si se utilizan los pines Tx y Rx tanto el Arduino como del ESP no podrían comunicarse con la Raspberry; y el servidor Rest y MQTT respectivamente.

Cuando el mensaje llegue al ESP8266 este lo decodificará y realizará la petición “GET” correspondiente a dicho elemento haciendo uso del servicio Rest mediante la URL.



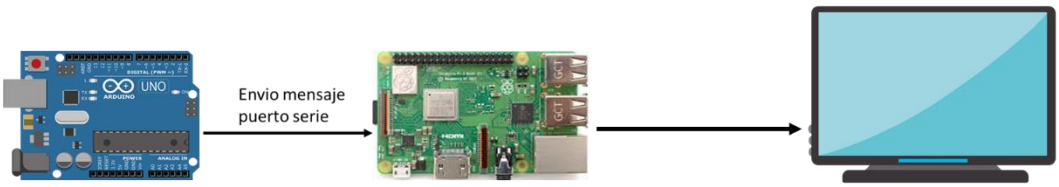
El servidor Rest recibirá dicha petición y accederá a la base de datos donde estará toda la información correspondiente a cada uno de los elementos. Esta información se la pasara al ESP8266 en formato json mediante MQTT.

Los parámetros recibidos del mensaje MQTT los imprimirá el LCD.



PROYECTO NUESTRA TABLA PERIÓDICA

Los videos estarán almacenados en la tarjeta SD de la Raspberry Pi donde accederemos mediante un script Python que lee del puerto serie los mensajes que le manda el Arduino y reproduce el video correspondiente.



Podrán verse y escucharse los videos en cualquier pantalla que tenga una conexión HDMI a la que pueda ser conectada la Raspberry Pi.

En principio la única funcionalidad que vamos a implementar es la de poder visualizar los videos del elemento seleccionado en un terminal remoto, en caso de que de tiempo nos gustaría añadir algunas ideas relacionadas con el mundo de la química, como por ejemplo un cuestionario interactivo con preguntas relacionadas con los elementos de la tabla, sus valencias, formulación…

Este proyecto tiene mucha visión de futuro, pero por falta de tiempo no podemos implementar todas las funciones que nos gustaría, ya que tenemos mas proyectos y asignaturas a parte de esta.

Análisis de la competencia:

Al ser un proyecto escolar pensado para un uso docente y presentarse en un concurso como una idea innovadora, no hemos encontrado ningún producto que sea similar.

Al no contar con una referencia real sobre el coste de dicho sistema presentamos a continuación una estimación del presupuesto necesitado para llevar a cabo dicho proyecto.

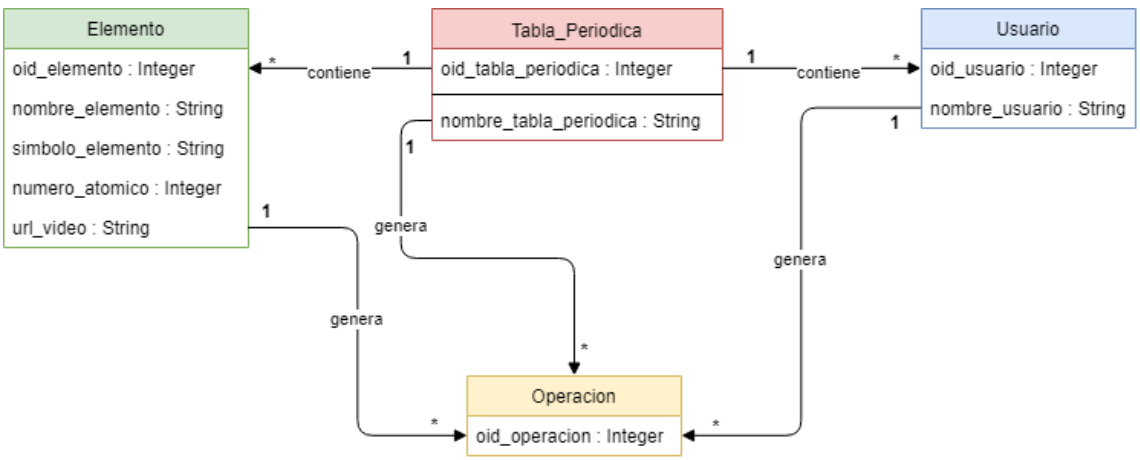
PROYECTO NUESTRA TABLA PERIÓDICA

Presupuesto:

1. Placa Arduino -> 14€ (link del producto) .
2. Placa ESP8266 -> 7’50€ (link del producto) .
3. 9 x Multiplexores CD74HC4067 -> 18’45€ (link del producto) .
4. Pantalla LCD -> 10€ (link del producto) .
5. 5 x Packs de 25 pulsadores -> 52,50€ (link del producto) .
6. 4 x Rollo cable 40x1m -> 32€ (link del producto) .
7. 2 x Pack de 3 protoboas -> 18’60€ (link del producto) .
8. Rasberry Pi 3 Model B+ -> 37’50€ (link del producto) .
9. Tarjeta SD 16 o 32 GB para Rasberry -> 4€ (link del producto) o 6€ (link del producto) .

Implementación Base de Datos:

Para realizar la implementación de la base de datos necesaria para nuestro proyecto hemos diseñado el siguiente diagrama UML a partir del cual se ha construido posteriormente la base de datos ya mencionada.



PROYECTO NUESTRA TABLA PERIÓDICA

Con motivo de que pudieran implementarse físicamente más de un sistema hemos optado por declarar una entidad “Tabla_Periodica” que contenga múltiples usuarios y elementos, de esta forma cada uno de los usuarios a la hora del registro podrá elegir el sistema físico con el que interactuar.

Una vez aclarado esto pasaremos a describir cada una de las entidades planteadas, así como la relación que guardan entre ellas.

Tabla_Periódica:

Es la base de toda la estructura planteada, representa cada uno de los sistemas que pudieran llegar a implementarse físicamente.

Guarda una relación 1-n con las entidades “Usuario”, “Elemento” y “Operación”, ya que una tabla periódica contiene varios usuarios, elementos y operaciones.

Usuario:

Referencia a cada uno de los usuarios que se registren mediante la aplicación web, estos estarán asociados a una única tabla periódica que se concretara en el momento del registro.

Guarda una relación 1-n con la entidad Operación, puesto que un usuario puede generar varias operaciones seleccionando múltiples elementos distintos pertenecientes a una tabla periódica.

Elemento:

Es la entidad que referencia a cada uno de los elementos correspondientes a la tabla periódica, cada uno de ellos tendrá asignado un botón físico con el que el usuario interactuará para generar las distintas operaciones.

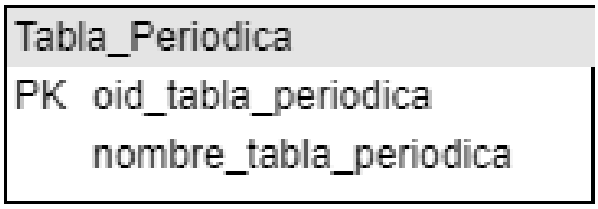
Tienen una relación 1-n con la entidad Operación ya que un usuario al seleccionar un elemento se pueden generar múltiples operaciones, asociadas tanto al usuario como al elemento.

PROYECTO NUESTRA TABLA PERIÓDICA

Operación:

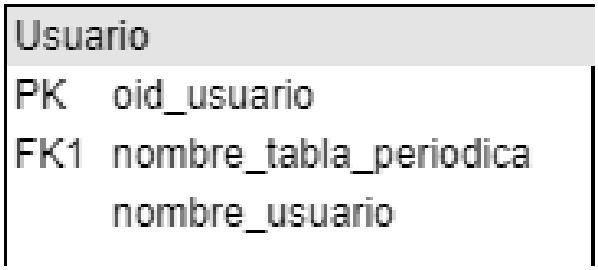
Es la entidad generada cuando un usuario asociado a una tabla periódica acciona unos de los botones correspondientes a cada uno de los elementos. De esta forma podremos controlar todas las interacciones que realice un usuario en una tabla y sobre los elementos que las realiza.

Ahora que comprendemos la estructura planteada en nuestra base de datos pasaremos a profundizar en cada una de las tablas implementadas.



“Tabla_Periodica” consta de una primary key de tipo Integer “oid_tabla_periodica” que se genera y autoincrementa automáticamente con cada tabla que se crea y nos sirve para identificar cada una de las tablas periodicas.

A parte hemos considerado oportuno introducir un parámetro de tipo String “nombre_tabla_periodica” que al igual que el “oid_tabla_periodica” sirve para identificar cada una de las tablas periódicas, pero este se ha creado con la intención de dar facilidades a nivel de usuario asignándole un nombre a cada una de ellas.

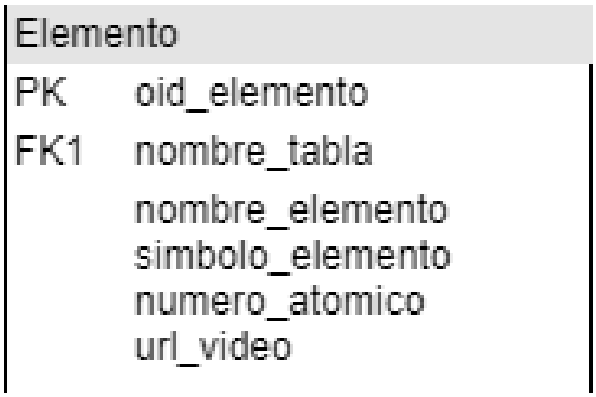


“Usuario” al igual que “Tabla_Periodica” tiene su primary key de tipo Integer autogenerada y autoincrementada que sirve para identificar al usuario.

PROYECTO NUESTRA TABLA PERIÓDICA

Como hemos visto anteriormente en el diagrama UML “Tabla_Periodica” y “Usuario” tienen una relación 1-n por lo que será necesario una foreing key “nombre_tabla_periodica” que servirá para asociar al usuario con la tabla periódica correspondiente.

Por último, constara de un parámetro “nombre_usuario” que será utilizado para que el propio usuario se registre con el nombre que guste en la aplicación web.



“Elemento” como las dos tablas anteriores consta de un identificador de tipo Integer con las mismas características que poseen los de las tablas anteriores “oid_elemento”.

Al igual que la tabla “Usuario” esta posee una relación 1-n con “Tabla_Periodica” por tanto será necesaria una foreing key de tipo String “nombre_tabla_periodica” para identificar a la tabla periodica a la que pertenece el elemento.

Todos los demás parámetros son las características de cada elemento como pueden ser “nombre_elemento” (nombre de cada uno - Hidrogeno), “símbolo_elemento” (símbolo correspondiente al elemento - H), el número atómico (número de protones que posee cada átomo del elemento - 1) y “url_video” (sirve para identificar el video explicativo del elemento correspondiente). Todos estos son únicos y no pueden repetirse.

Operacion	
PK	oid_operacion
FK1	nombre_tabla
FK2	nombre_usuario
FK3	nombre_elemento

Y por último la tabla “Operación” estará compuesta como todas las tablas anteriores de una primary key de tipo Integer “oid_operacion” que tendrá la misma función y características que las anteriores.

Al tener una relación 1-n con cada una de las tablas anteriores (“Tabla_Periodica”, “Usuario”, “Elemento”) esta vez serán necesarias 3 foreing keys de tipo String que identifiquen respectivamente a cada una de las tablas referenciadas.

• Servicio REST:

El servicio Rest es un servidor en el que existen una serie de métodos implementados que nos ayudarán a mostrar la información contenida en la base de datos.

Inicialmente, en el servicio Rest se realiza la conexión con la base de datos, mediante el uso de la sentencia mySQLClient que contendrá el verticle desplegado y una serie de configuraciones, entre las cuales se encuentra el host, que en este caso sería localhost, un usuario, una contraseña, el nombre de la base de datos y el puerto en el que escucha.

PROYECTO NUESTRA TABLA PERIÓDICA

Métodos GET:

1. handlerGetTablaPeriodica();
   Éste método es el encargado de tomar toda la información de la base de datos referente a todas las tablas “TablaPeriodica”, donde en ella se encuentra un identificador “oid_tabla_periodica”, de la propia tabla que será un tipo INT y de manera autoincrementar por ser la PK de la tabla. Y “nombre_tabla_periodica”, que como se explicó anteriormente, sirve para que el usuario tome la decisión de escoger la tabla física en la que quiere interactuar, siendo ésta de tipo String y un valor único, es decir, que no se pueda repetir.
   Para ello creamos una lista vacía de tipo TablaPeriodica(Entitie), se recorrerá mediante un for incluyendo todos los Json que se reciben. Además, hemos implementado una serie de sentencias que gestionan los errores cometidos, así como el cierre de conexión tras obtener el resultado.
2. handlerGetTablaPeriodicaNombre();
   Este método es un extendido del anterior, es decir, realiza exactamente lo mismo que handlerGetTablaPeriodica(), pero aquí aplicamos un filtro que consiste en mostrar toda la información de una sola tabla periódica correspondiente al “nombre_tabla_periodica” que especificamos en la URL.

3. handlerGetTodosUsuarios();
   Este GET toma toda la información acerca de todos los usuarios desde la tabla “Usuarios”, donde se encuentra un “oid_usuario” de tipo INT y de manera autoincremental por ser la PK de esta tabla. También un “nombre_usuario” de tipo String y único, mediante el cual un usuario se registrará en la aplicación web con ese nombre y se guardará en la base de datos. Además, hay un “nombre_tabla_periodica” de tipo String y único, que se especifica el nombre de la tabla física con la que se interactuará y se elegirá también en la aplicación web.
   Como explicamos en el primer GET, todos realizan lo mismo, una lista que se recorre y se guarda los Json que se reciben.

4. handlerGetUsuarioNombre();
   Este es el extendido de handlerGetTodosUsuario(), en el que se le aplica el filtro por un "nombre_usuario" especificado en la URL, y mostrará la información de un solo usuario registrado.
5. handlerGetTodosElementos();

   Este método GET toma toda la información relacionada con todos los elementos contenidos de una tabla periódica, es decir, mostrará información de todos los elementos. Contiene un "oid_elemento" de tipo INT y de manera autoincremental al ser la PK de esta tabla. También un "nombre_elemento" de tipo String, que hará referencia al nombre del elemento que busquemos, un "símbolo_elemento" de tipo String que hace referencia a su símbolo, un "numero_atomico" de tipo INT que tiene el valor del peso atómico del elemento, una "url_video" de tipo String que será una url del video que se mostrará por pantalla al elegir un elemento específico y un "nombre_tabla_periodica" de tipo String que hará referencia al nombre de la tabla física.

   El procedimiento es el mismo que los anteriores.

6. handlerGetElementoNombre();

   Este método realiza filtro de los elementos pasándole un "nombre_elemento" a la URL, así escogerá solo el elemento solicitado.

7. handlerGetTodasOperaciones();

   Este método recoge información sobre todas las operaciones realizadas en la tabla periódica, es decir, sobre los elementos pulsados en la tabla. Contiene un "oid_operacion" de tipo INT y de manera autoincremental por ser PK de la tabla. También un "nombre_tabla_periódica" de tipo String, que como hemos explicado antes, sirve para referenciar la tabla física. También un "nombre_elemento" de tipo String que será el elemento que solicitemos al pulsar un botón específico. Y por último un "nombre_usuario" de tipo String que se registrará quien ha realizado dicha operación.

El procedimiento es el mismo que los anteriores.

8. handlerGetOperacionesNombreUsuario();

Mediante este método, recogemos la información de una sola operación, pasándole como filtro un “nombre_usuario” en la URL, para determinar las operaciones que ha realizado un determinado usuario.

Métodos PUT/POST:

9. handlerInsertTablaPeriodica();

Método implementado para realizar los inserts de la tabla “Tabla_Periodica” en la base de datos con el correspondiente cuerpo:

```
{
    "nombre_tabla_periodica" : "ejemplo"
}
```

10. handlerInsertUsuario();

Método implementado para realizar los inserts de la tabla “Usuario” en la base de datos con el correspondiente cuerpo:

```
{
    "nombre_usuario" : "juan" ,
    "nombre_tabla_periodica" : "tabla01"
}
```

11. handlerInsertElemento();

Método implementado para realizar los inserts de la tabla “Elemento” en la base de datos con el correspondiente cuerpo:

```
{
    "nombre_elemento" : "Hidrogeno" ,
    "simbolo_elemento" : "H" ,
    "numero_atomico" : 1 ,
    "url_video" : "hidrogeno.avi" ,
    "nombre_tabla_periodica" : "tabla01"
}
```

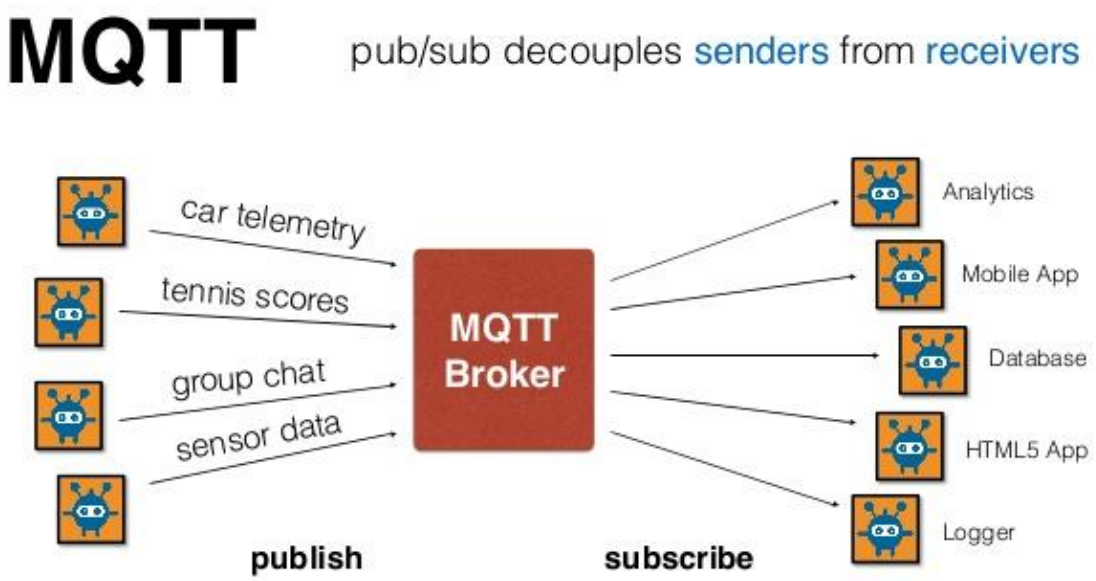
12. handlerInsertOperacion();

Método implementado para realizar los inserts de la tabla "Operacion" en la base de datos con el correspondiente cuerpo:

```
{
    "nombre_tabla_periodica" : "tabla01",
    "nombre_usuario" : "juan" ,
    "nombre_elemento" : "Hidrogeno"
}
```

- MQTT:

Es una norma protocolo de mensajería basado en publicación-suscripción. Funciona sobre el protocolo TCP / IP. Está diseñado para conexiones con ubicaciones remotas donde el ancho de banda de la red es limitado. El patrón de mensajería de publicación-suscripción requiere un intermediario de mensajes, denominado como Broker.



PROYECTO NUESTRA TABLA PERIÓDICA

En nuestro caso, el cliente será la placa ESP8266 Node MCU v1.0 y enviará la información a la pantalla LCD. El Broker mandará una publicación cada 2 segundos.

1. void makeGetRequestElemento(const char* nombreElemento);

Función que realiza una petición “GET" a la tabla “elemento” de la base de datos mediante una URL especificada al servidor REST(https://localhost:8090/elemento/nombreElemento).

Básicamente los parámetros que recibe son:

- oid_elemento
- nombre_elemento
- simboloElemento
- numeroAtomico
- url_video
- nombre_tabla_periodica
- El nombre del elemento se le pasará por parámetros, para tomarlo directamente del Array de elementos que se creó e inicializó con todos los elementos de la tabla para realizar la decodificación del mensaje que se recibe del puerto serie del Arduino.

Como respuesta a nuestra petición se generará un JSON que será enviado mediante un mensaje utilizando MQTT al ESP8266 que a su vez se comunicará con el LCD que recogerá los parámetros convenientes y los mostrará por pantalla. Los datos a imprimir serán los siguientes:

nombreElemento – simboloElemento – numeroAtomico

En éste método también se lleva a cabo un control de errores, es decir, una vez que se realiza la petición “GET”, pintaremos por el puerto serie del ESP8266 el código de respuesta de ésta.

Si recibimos un 200 significa que todo ha ido bien, si es un 400 significa que podría no haber encontrado el dato que se busca y si por el contrario se obtiene un número menor que 0, significa que se habrá producido un error en la petición.

2. void makePutRequestOperacion(const char* nombreElemento);

Función que realiza un “PUT” en la tabla “operación” de la base de datos, recogiendo los datos que nos llega por el mensaje MQTT. Manda los siguientes datos:

- nombre_tabla_periodica, como en principio solo hay una hemos dejado por defecto el nombre de la tabla periódica principal “tabla01”.
- nombre_usuario, nombre del usuario que está realizando la operación correspondiente.
- nombre_elemento, nombre del elemento al que se haya realizado la petición “GET” anterior a esta petición.

Con esta petición tendremos un control de los elementos a los que se ha accedido, desde que tabla periódica han sido accedidos y que usuario ha sido el que ha registrado la operación.

Una vez realizado el “PUT”, como en la función anterior, también se realiza un control de errores.

• Python:

Para poder reproducir el video correspondiente al elemento accedido hemos desarrollado un script en Python que ejecutamos en la Raspberry Pi. El procedimiento que sigue es el siguiente:

1. Lee el puerto serie, donde le están llegando mensajes del Arduino que al igual que el ESP8266 tendrá que decodificarlo para ver de que elemento se trata y así reproducir el video correspondiente. Esta lectura tendrá que ser realizada continuamente.

```
PuertoSerie = serial.Serial('/dev/ttyACM0',115200)

while True:
    arduino = PuertoSerie.readline()
```

2. Una vez hemos recibido el mensaje del Arduino por el puerto serie tendremos que identificar el elemento que queremos acceder.

```
if arduino.rstrip()=='0':
```

3. Por último, cuando hemos identificado de que elemento se trata creamos un subproceso mediante la consola que crea una instancia del reproductor vlc que accede a una ruta especificada y realiza la reproducción.

```
#player1.play()
subprocess.call(['vlc' ,'-f' ,'./Downloads/video1.mp4'])
```

- ## Cambios Realizados:

Al tener un limite de tiempo de 4 meses para realizar este proyecto algunas de los elementos han tenido que ser eliminados, al igual que otros que han sido sustituidos por alguna otra metodología que nos ha permitido realizar la misma funcionalidad, pero dentro del límite establecido. Algunos de estos cambios y modificaciones han sido las siguientes:

1. Aplicación web/movil:

   En primer lugar, hemos aparcado la opción del desarrollo de una aplicación web para la reproducción de videos ya que encontramos que a través de Python podíamos realizar esta acción y era más asequible en términos de tiempo.

   Aun habiendo solventado este problema desarrollando el script hay algunas limitaciones como que vlc tiene un buffer que se bloquea y no permite parar un video para reproducir otro al pulsar otro botón distinto.

2. Eliminación del altavoz:

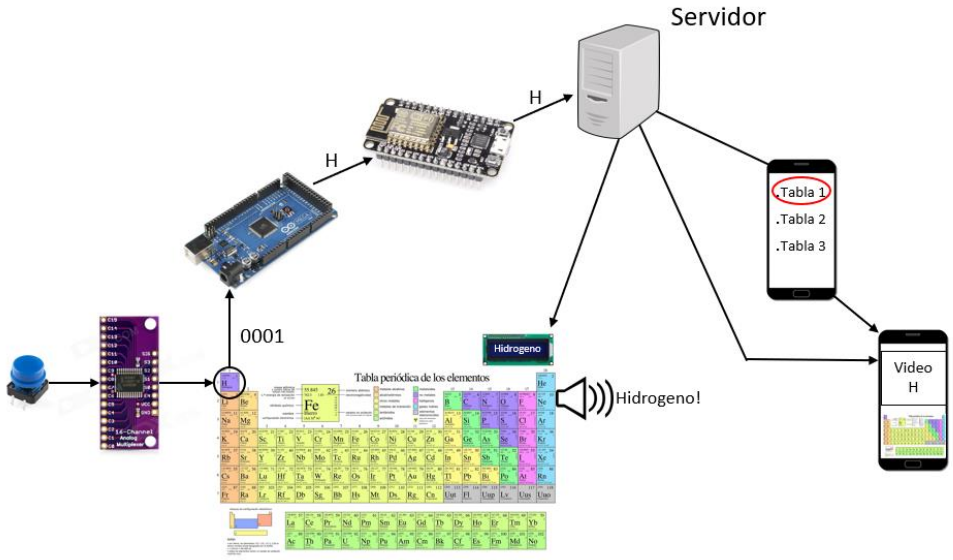
   Con la realización del script anterior los videos son emitidos a través del HDMI por lo que no ha sido necesario implementar un altavoz ya que esta interfaz transmite audio, por lo que si se conecta a cualquier terminal con altavoces este se escuchara correctamente.

PROYECTO NUESTRA TABLA PERIÓDICA

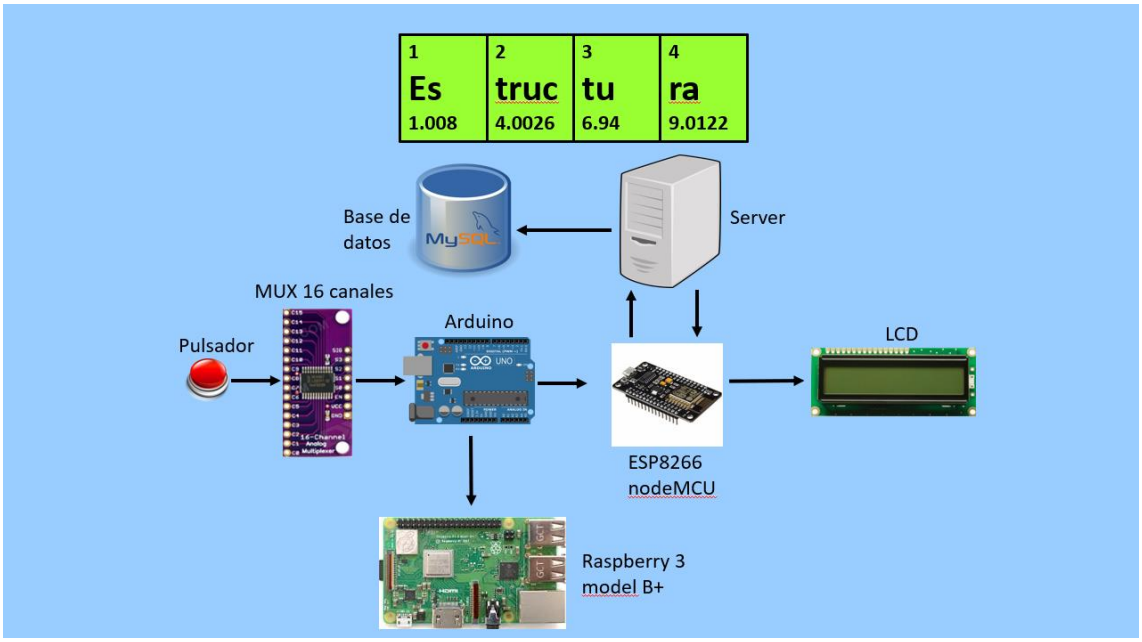
3. Estructura básica del proyecto:

Al haber realizado estas modificaciones anteriores el esquema del proyecto ha cambiado ligeramente.

Estructura inicial del proyecto:



Estructura final:



Como se puede observar se ha eliminado el altavoz y la aplicación web/móvil planteada en un primer momento; y se ha incluido la Raspberry Pi que como hemos visto antes cumple con el propósito requerido.

PROYECTO NUESTRA TABLA PERIÓDICA

- ## Conclusiones:

A priori este proyecto parecía simple, pero a medida que hemos ido avanzando a lo largo del curso y hemos ido implementando nuevos protocolos como MQTT han ido surgiendo dudas y fallos que hemos ido solucionando poco a poco.

El mayor problema que hemos tenido ha sido a nivel de hardware ya que hay que configurar muchos botones y componentes para su correcto funcionamiento. Al igual que a nivel de software donde MQTT ha sido un poco problemático ya que no entendíamos la metodología que seguía.

A pesar de todas las dificultades estamos muy satisfechos con el resultado obtenido y seguiremos trabajando en el proyecto para poder perfeccionarlo e implementarlo en el centro con el que estamos colaborando.

- ## Video del montaje y correcto funcionamiento del proyecto:

https://drive.google.com/open?id=1jkB_NKzkYO7NfYCLMBxCTWZE6TcRv6sF