ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA TELEMÁTICA

DESARROLLO DE UNA PLATAFORMA IOT PARA CONOCER EL MAPA SONORO AMBIENTAL DE UN ENTORNO EXTERIOR

Autor: Conchita Martín Velázquez-Gaztelu

Director: Miguel Ángel Sanz Bobi

Madrid

Julio 2019

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Desarrollo de una plataforma de IoT para conocer el mapa sonoro ambiental de un entorno exterior

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico **2018/19** es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Conchita Martín Velázquez-Gaztelu Fecha: 12/ 07/ 2019

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Miguel Ángel Sanz Bobi Fecha: 12/ 07/ 2019

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor Conchita Martín Velázquez-Gaztelu

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: Desarrollo de una plataforma de IoT para conocer el mapa sonoro ambiental de un entorno exterior, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente)*.

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que

pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 12. de Julio de 2019

ACEPTA

Fdo……

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA TELEMÁTICA

DESARROLLO DE UNA PLATAFORMA IOT PARA CONOCER EL MAPA SONORO AMBIENTAL DE UN ENTORNO EXTERIOR

Autor: Conchita Martín Velázquez-Gaztelu

Director: Miguel Ángel Sanz Bobi

Madrid

Julio 2019

Agradecimientos

Quisiera agradecer a varias personas la ayuda que me han prestado directa o indirectamente en la elaboración del proyecto de fin de grado.

En primer lugar, como no podría ser de otra manera, a mi director Miguel Ángel, por todo el tiempo dedicado en este proyecto y por la ilusión que ha puesto desde el primer momento.

A mis compañeros de clase Irene, Carlos, Luis, Esther, Laura y Javi que tanta ayuda me han ofrecido y con los que mantendré siempre una buena amistad. Pero en especial a Marta, mi compañera y buena amiga, que ha estado conmigo 5 años compartiendo aula y que tanto cariño me ha dado. Terminar el proyecto y el grado no hubiera sido posible sin su ayuda.

A mi familia, por todo el apoyo que he recibido. Han sido los que me han animado a llegar hasta aquí y los que han hecho un gran esfuerzo tanto económica como personalmente para que pueda ser ingeniera. En especial mi hermano Jacobo, que me manda fuerza desde el cielo.

Y por último, a mi novio. La persona que cada día me anima, me motiva y la que mejor me aconseja. Es el mejor ejemplo de cómo ser un buen ingeniero y luchar por lo que quieres.

A todos, gracias.

DESARROLLO DE UNA PLATAFORMA IOT PARA CONOCER EL MAPA SONORO AMBIENTAL DE UN ENTORNO EXTERIOR

Autor: Martín Velázquez-Gaztelu, Conchita

Director: Sanz Bobi, Miguel Ángel

Entidad Colaboradora: ICAI – Universidad Pontificia de Comillas

RESUMEN DEL PROYECTO

En el presente proyecto se ha realizado un equipo de sonido para elaborar un mapa autoorganizado con el fin de encontrar patrones en un entorno exterior. Para ello se ha utilizado un hardware utilizando *Arduino* de 3 sonómetros para recoger los datos sonoros y se ha desarrollado en *Matlab* un código para crear una red neuronal.

Palabras clave: SOM, patrones sonoros, mapa sonoro ambiental, sonómetro, red neuronal.

1. Introducción

La contaminación acústica es hoy en día uno de los problemas a tratar en las grandes ciudades. Para impedir los problemas que supone vivir con altos niveles de ruido, se hacen estudios de contaminación acústica por todas las ciudades, y se instalan sonómetros en locales donde frecuenta mucha gente para tener un control sobre dichos niveles.

El objetivo de este proyecto es pues, desarrollar un equipo de medida de sonido ambiental de bajo coste, para poder instalarlo donde se desee. Este equipo tendrá comunicación wifi con un sistema de almacenamiento desde donde analizar patrones sonoros de un entorno exterior en el que obtener sus patrones o mapas de sonido.

2. Definición del Proyecto

La primera etapa del proyecto consiste en la búsqueda de materiales para el equipo de sonido. Se hace una comparación de precios y características y se determinan qué tecnologías son las más apropiadas para utilizar en el proyecto.

En la segunda etapa de este proyecto se hace el diseño del equipo con los elementos elegidos en la anterior etapa y se implementa el sonómetro.

A continuación, comprueba el funcionamiento del sonómetro. Esta etapa consiste en 3 fases en las que se irá comprobando el funcionamiento del sonómetro y el funcionamiento del wifi de la placa. Una vez que el sonómetro funciona correctamente, se configuran otros 2 sonómetros para tener la red preparada para la recogida de datos.

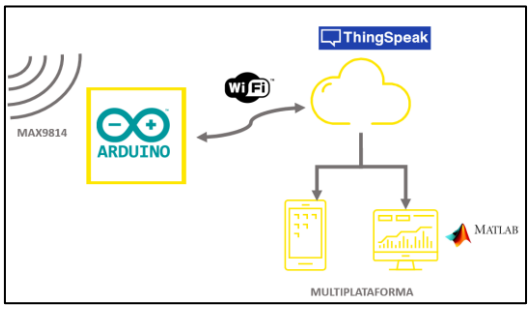


Ilustración 1. Esquema Alto Nivel.

La última etapa del proyecto es crear el mapa de sonido. En primer lugar, se hace la recogida de datos (se recogen en total 8.400 muestras) para poder hacer la fase de entrenamiento del mapa. El objetivo del mapa es poder distinguir entre franja horaria y el día de la semana. Una vez que se construye el mapa, se pasa a la fase de verificación y se sacan las conclusiones.

3. Descripción del modelo/sistema/herramienta

Para la realización del proyecto se ha diseñado un equipo de medida de sonido compuesto por un sonómetro MAX 9814 y por una placa *Arduino YUN*.

La herramienta que se han usado para crear la red neuronal es *Matlab*. Gracias a esta herramienta se ha conseguido crear un SOM con 25 patrones para posteriormente sacar conclusiones del entorno sonoro en concreto.

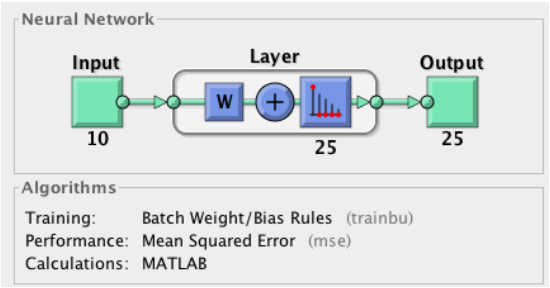


Ilustración 2. Arquitectura SOM del proyecto.

4. Resultados

Se ha realizado un SOM con 840 muestras de las 8.400 muestras recogidas. Este mapa ha sido creado para 25 patrones de sonido.

En la fase de verificación han salido los siguientes resultados:

- Ningún patrón tiene un margen de error de más del 6%.
- 19 de los 25 patrones de sonido tienen un margen de error menor a 1%.
- Tan solo 6 patrones de sonido tienen un margen de error de entre 1-6 %.

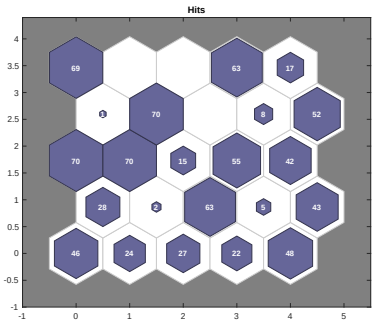


Ilustración 3. Estructura de la red neuronal.

5. Conclusiones

Con los resultados obtenidos en la fase de verificación se puede observar que los patrones son muy exactos. Esto lleva a la conclusión que se puede identificar un patrón para un entorno en concreto, y obtener qué día de la semana y qué franja horaria se está escuchando desde una muestra de datos.

DEVELOPMENT OF AN IOT PLATFORM TO GET TO KNOW THE ENVIRONMENTAL SOUND MAP OF AN OUTDOOR ENVIRONMENT

Author: Martín Velázquez-Gaztelu, Conchita

Supervisor: Sanz Bobi, Miguel Ángel

Collaborating Entity: ICAI – Universidad Pontificia de Comillas

ABSTRACT

In the present project a sound system has been made to elaborate a self-organized map in order to find patterns in an external environment. For this purpose, an *Arduino* hardware with three sound level meters has been used to collect the sound data and a code has been developed in *Matlab* to create a neural network.

Keywords: SOM, sound patterns, environmental sound map, sound level meter, neural network.

1. Introduction

Noise pollution is today one of the problems to be dealt with in big cities. In order to prevent the problems of living with high noise levels, noise pollution studies are carried out in all cities, and sound level meters are installed in places where many people frequent in order to have control over these levels.

The aim of this project is therefore to develop a low-cost environmental sound measurement equipment, so that it can be installed wherever it is desired. This equipment will have wifi communication with a storage system from where to analyze sound patterns of an external environment in which to obtain their patterns or sound maps.

2. Definition of the project

The first stage of the project consists of the search for materials for the sound equipment. A comparison of prices and characteristics is made and it is determined which technologies are the most appropriate to use in the project.

In the second stage of this project, the design of the equipment is done with the elements chosen in the previous stage, and the sound level meter is implemented.

Then, the sound level meter is checked. This stage consists of 3 phases, in which the operation of the sound level meter and the operation of the wifi of the board will be checked. Once the sound level meter works correctly, another 2 sound level meters are configured to have the network ready for data collection.

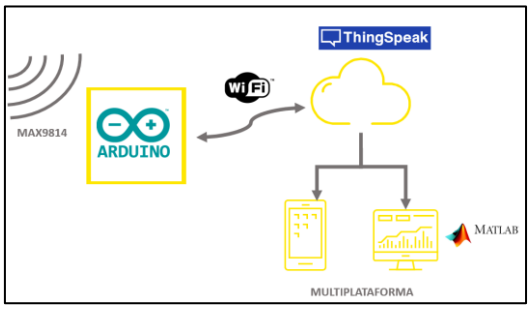


Figure 4. Scheme of the project.

The last stage of the project is to create the sound map. First of all, the data is collected (a total of 8,400 samples are collected) to be able to do the training phase of the map. The objective of the map is to be able to distinguish between the time zone and the day of the week. Once the map is constructed, the verification phase begins and the conclusions are drawn.

3. Description of the model/system/tool

For the realization of the project has been designed a sound measuring equipment consisting of a sound level meter MAX 9814 and an Arduino YUN board.

The tool used to create the neural network is Matlab. Thanks to this tool it has been possible to create a SOM with 25 patterns to later draw conclusions from the specific sound environment.

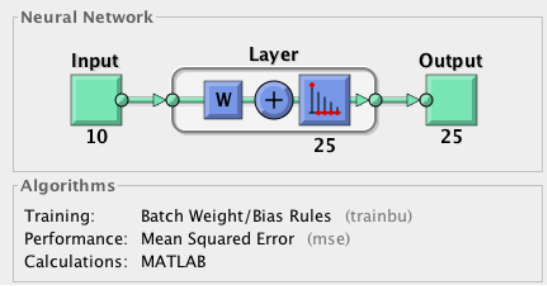


Figure 5. SOM Architecture.

4. Results

An SOM has been carried out with 840 samples from the 8,400 samples collected. This map has been created for twenty-five sound patterns.

In the verification phase, the following results were obtained:

- No pattern has a margin of error of more than 6%.
- 19 of the 25 sound patterns have an error margin of less than 1%.
- Only 6 sound patterns have a margin of error between 1-6%.

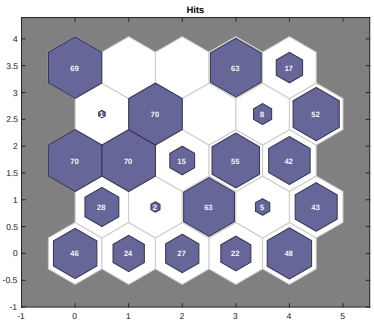


Figure 6. Structure of the neuronal network.

5. Conclusions

With the results obtained in the verification phase it can be observed that the patterns are very exact. This leads to the conclusion that you can identify a pattern for a particular environment, and get which day of the week and which time zone you are listening to from a data sample.

Índice de la memoria

Índice de figuras

Índice de tablas

Glosario

IoT (Internet of Things): En español, internet de las cosas. Es un concepto que se refiere a la interconexión digital de objetos cotidianos con internet. [1]

Sonómetro: Instrumento que sirve para medir y comparar sonidos.

M2M (Machine to Machine): En español, máquina a máquina. Es un concepto de comunicación entre dos máquinas.

Equipo: En este proyecto, se habla de equipo cuando se refiere a un sistema capaz de enviar y recibir datos.

Matlab: es una herramienta de cálculo numérico que ofrece un entorno de desarrollo integrado muy útil para operaciones numéricas. Entre los usos de esta aplicación, se destaca para este proyecto, la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos y la comunicación con otros programas. [2]

Mapa Autoorganizado (SOM): Se explica en detalle en el ***apartado 3.2 Mapa Autoorganizado***. Es un tipo de red neuronal artificial que es entrenada para producir una representación discreta del espacio de las muestras de entrada.

Capítulo 1. INTRODUCCIÓN

La contaminación acústica es hoy en día uno de los problemas a tratar en las grandes ciudades. Al ser las ciudades cada vez más grandes, la actividad humana como el transporte, construcción o la industria aumenta, provocando grandes problemas para la población. Esto puede suponer un peligro ya que la salud auditiva está muy relacionada con problemas de insomnio, depresión, de memoria, de atención… pudiendo causar daños irreversibles.

Para impedir estos problemas, se hacen estudios de contaminación acústica por todas las ciudades, y se instalan sonómetros en locales donde frecuenta mucha gente para tener un control sobre dichos niveles.

Además, desde hace algunos años, las grandes ciudades europeas están obligadas a elaborar mapas de ruido. [3]

El objetivo de este proyecto es pues, desarrollar un **equipo de medida de sonido ambiental** de bajo coste, para poder instalarlo donde se desee. Este equipo tendrá comunicación wifi con un sistema de almacenamiento desde donde analizar patrones sonoros de un entorno exterior en el que obtener sus patrones o mapas de sonido.

1.1 MOTIVACIÓN DEL PROYECTO

La principal motivación de este proyecto es medir los niveles acústicos de cualquier entorno de una forma fácil y económica con un equipo sencillo y de tamaño muy reducido, que se puede implantar en cualquier espacio, y ampliar el número de unidades sensoras según se requiera.

El sistema puede ser implantado en numerosos sitios según el uso. Algunos de los ejemplos son:

- Locales de noche, para controlar el nivel de ruido de la música.
- Universidades, para controlar el nivel de ruido de las clases.
- En espacios privados, para hacer un control de anti-intrusiones.
- Viviendas cerca de aeropuertos o grandes autovías.

1.2 ESTADO DEL ARTE

Hoy en día existen muchos métodos de clasificación de datos, relacionados con analizar patrones. A continuación, se explican algunos de los trabajos encontrados sobre aplicaciones con mapas autoorganizados.

Visualización del estado de ánimo de la música mediante mapas autoorganizados [4]

Se han creado sistemas automáticos de organización de la música. Este trabajo en concreto presenta una aproximación a la representación gráfica del estado de ánimo de canciones basadas en mapas autoorganizados.

Aplicación de los mapas autoorganizados en el sector marítimo [5]

Actualmente se están desarrollando sistemas que dan solución a los problemas que presentan un entorno como el mar. Los datos que se recogen de un satélite sobre una zona de mar concreta son muy parecidos entre sí y es difícil detectar el lugar exacto. Analizar un entorno sonoro en el mar, puede dar información sobre previsión de tiempo, distancia a la costa, …

Monitoreo de sonido ambiental dependiente del contexto usando SOM junto con LEGION [6]

Las redes de medición de ruido ambiental se aplican cada vez más para el seguimiento de la contaminación acústica en un contexto urbano. Los nodos de medición inteligentes ofrecen la oportunidad de realizar un análisis avanzado del sonido ambiental, pero aún es complicado el factor coste y funcionalidad.

Cuando se utiliza una arquitectura escalonada, se pueden utilizar nodos locales con capacidades de computación limitadas para detectar eventos sonoros de interés potencial, que luego son analizados por nodos más potentes. Hay presentes modelos de imitación humana para detectar eventos sonoros raros y ruidosos.

Capítulo 2. DEFINICIÓN DEL TRABAJO

2.1 JUSTIFICACIÓN

Como se ha dicho anteriormente, desde hace algunos años, las grandes ciudades europeas están obligadas a elaborar mapas de ruido. [3] Sin embargo, los modelos utilizados se basan en aproximaciones, implican un tiempo de cálculo significativo y los datos de entrada son difíciles de adquirir.

Todas estas limitaciones han llevado a una falta de realismo. Además, "la evaluación de la percepción de los usuarios del entorno sonoro no se tiene en cuenta en estos mapas", explica Erwan Bocher. Sin embargo, se trata de un criterio importante, ya que el ruido medido objetivo no transmite necesariamente el factor de molestia real.

Es por este motivo por lo que en este proyecto se estudia la manera de interpretar un entorno sonoro exterior en mapas autoorganizados.

Al ser un equipo de medida de bajo coste, puede permitir que esté a disposición del público y ayudar a autoridades a establecer planes de acción.

2.2 OBJETIVOS

El objetivo principal del proyecto es el desarrollo de un equipo de medida de sonido local ambiental de bajo coste con comunicación wifi que permita enviar los datos recogidos a un puesto central de análisis en donde hay un sistema de almacenamiento. Con esta información, un programa inteligente realiza el descubrimiento de patrones sonoros del entorno exterior, objeto de análisis. Estos mapas de sonido permitirán ayudar a tomar decisiones sobre su posible impacto tanto en las actividades de su entorno que puedan

verse afectadas como la planificación de otras en periodos donde se prevé un menor impacto sonoro exterior.

2.3 METODOLOGÍA

El proyecto queda dividido en 5 etapas:

Figura 1. Etapas y fases del proyecto.

Para la realización del proyecto se ha utilizado una metodología de desarrollo de software tradicional (análisis y diseño, implementación, pruebas), que se ha seguido durante toda la vida del proyecto, añadiendo las conclusiones finales. Se pueden ver más detalles de la metodología en el ***apartado 2.4 Planificación y Estimación económica***.

En los siguientes apartados se hace una descripción de las fases más relevantes.

2.3.1 ETAPA 0

Esta etapa se ha definido con el tutor del proyecto y es cuando se dejan claros los requisitos y el objetivo final del proyecto.

2.3.2 ETAPA 1

La primera etapa del proyecto consiste en la búsqueda de materiales para el equipo de sonido. Se hace una comparación de precios y características y se determinan qué tecnologías son las más apropiadas para utilizar en el proyecto.

2.3.3 ETAPA 2

En la segunda etapa de este proyecto se hace el diseño del equipo con los elementos elegidos en la anterior etapa y se implementa el sonómetro.

2.3.4 ETAPA 3

En esta etapa se comprueba el funcionamiento del sonómetro. Esta etapa consiste en 3 fases en las que se irá comprobando el funcionamiento del sonómetro y el funcionamiento del wifi de la placa.

2.3.5 ETAPA 4

La etapa 4 consiste en repetir la etapa 3, en otros dos sonómetros, para tener la red de los 3 sonómetros.

2.3.6 ETAPA 5

La última etapa del proyecto es crear el mapa de sonido. Esta etapa es la más larga del proyecto ya que es la más compleja.

En primer lugar, se hace la recogida de datos (de muchas muestras), para poder hacer la fase de entrenamiento del mapa.

Una vez que se construye el mapa, se pasa a la fase de verificación y se sacan las conclusiones.

2.3.7 CONCLUSIONES

Se realiza una extracción de conclusiones sobre los resultados obtenidos.

2.4 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

La planificación temporal del proyecto es la siguiente:

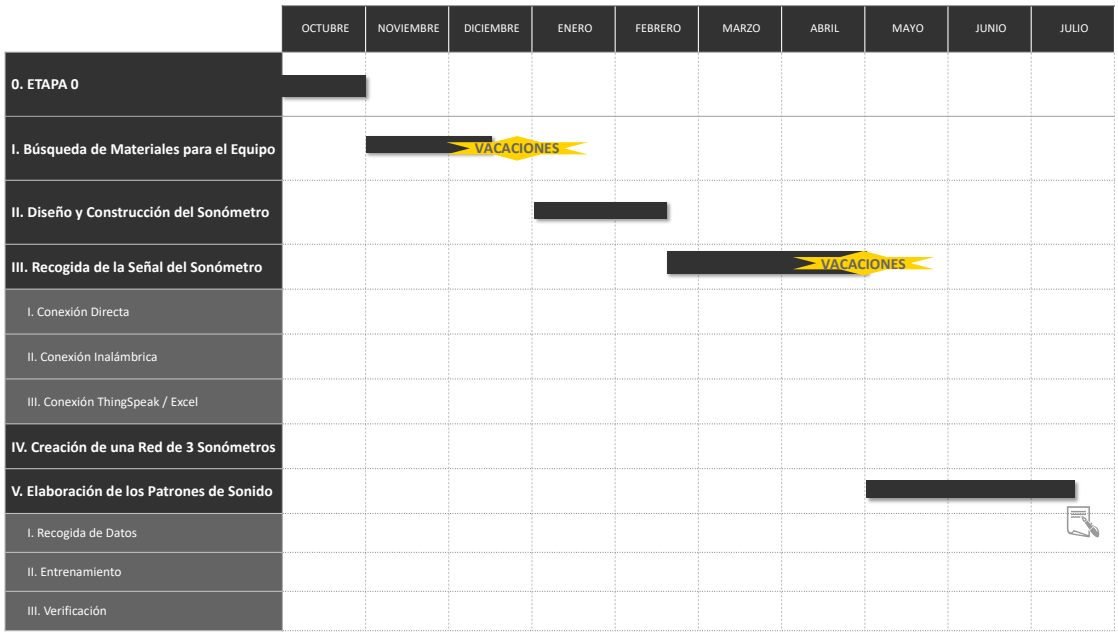


Figura 2. Planificación del proyecto.

Se ha tenido en cuenta para la elaboración del proyecto el periodo de vacaciones de Navidad y Semana Santa.

Desde el punto de vista económico se ha estimado el siguiente coste:

Gestión del proyecto	10.000,00 €
Sonómetros (x 3)	15,00 €
Arduino YUN (x 3)	400,00 €
Licencia Matlab	800,00 €
Baterías	200,00 €
Total	11.415,00 €

Tabla 1. Estimación económica.

Capítulo 3. DESCRIPCIÓN DE LOS MAPAS AUTOORGANIZADOS

En este capítulo se explican los conceptos más importantes de la última etapa (***Etapa V. Elaboración de Patrones de Sonido***), que serán de los que saquemos conclusiones.

3.1 REDES NEURONALES

Las redes neuronales artificiales (RNA) son un modelo computacional vagamente inspirado en el comportamiento observado de las redes neuronales biológicas. [7] Esta tecnología es el tipo de inteligencia artificial más sencillo ya que las neuronas son capaces de crear su propia representación tras una fase de entrenamiento.

Estas redes están formadas por un conjunto de unidades de datos llamados nodos o neuronas que están conectadas entre sí para transmitirse señales. Estos nodos tienen tanto entradas como salidas y está organizados por capas.

3.1.1 FASE DE ENTRENAMIENTO

Una vez hecha la estructura de la red neuronal, se pasa a la fase de entrenamiento. Esta fase consiste en insertar patrones objetivo a las entradas de las neuronas, y así los valores de los nodos se ajustan de forma iterativa hasta llegar a respuestas satisfactorias. [8]

Al final de la fase de entrenamiento, la red sabrá cuando crear, destruir y modificar nodos para dar respuestas óptimas al conjunto de patrones de entrenamiento.

3.1.2 TIPOS DE APRENDIZAJE

Existen varios tipos de aprendizaje de las RNA:

- Aprendizaje Supervisado: La red dispone [8] de unos patrones de entrada y salida que queremos obtener para una cierta entrada de información y en función de ello, se modifican los nodos ocultos para ajustar la entrada y la salida. Por lo cual, la red será capaz de predecir el valor correspondiente a cualquier tipo de entrada después de haber visto un par de ejemplos.
- Aprendizaje No Supervisado: Este aprendizaje consiste en no proporcionar a la red los patrones de salida sino solo los de entrada y dejar que la red los clasifique en función de características comunes que encuentre entre ellos. La red que se usará en este trabajo pertenece a esta categoría.

3.2 MAPA AUTOORGANIZADO

Un **mapa autoorganizado** (**S**elf-**O**rganizing **M**ap, en adelante SOM) es un tipo de red neuronal artificial que es entrenada usando **aprendizaje no supervisado** para producir una representación discreta del espacio de las muestras de entrada, llamado mapa [9].

El objetivo de este tipo de aprendizaje es categorizar los datos que se introducen a la red. Se clasifican valores similares en la misma categoría y, por tanto, deben activar la misma neurona de salida. [10]

3.2.1 FUNCIONAMIENTO

Los SOMs están compuestos por dos capas de neuronas, una de entrada y otra de salida. Cada neurona de los mapas SOM tiene asociado un vector de pesos que define su posición en el mapa.

Dos conceptos importantes en cuanto al funcionamiento de los SOMs son vecindad y competencia.

Los mapas autoorganizados utilizan la función de vecindad para poder organizarse y preservar las propiedades del espacio de entrada. Por lo cual, en las neuronas de la capa de salida habrá cierta influencia entre las vecinas.

Además, las neuronas también se organizan con una competición interna según su distancia (explicación más adelante en el ***apartado 3.2.3. Algoritmo***).

3.2.2 ARQUITECTURA

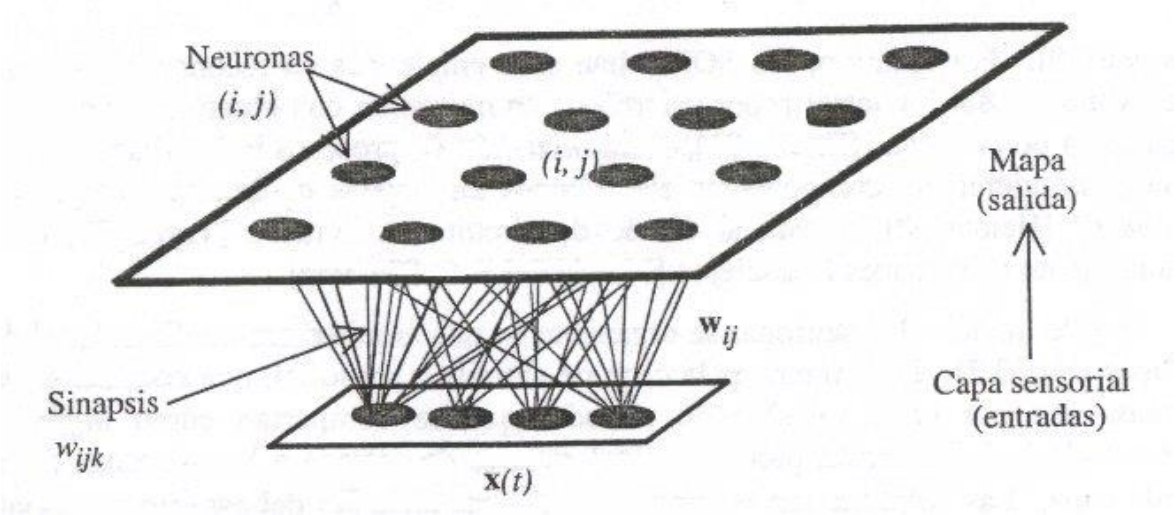


Figura 3. Arquitectura típica de un SOM. [11]

En la figura anterior se muestra la arquitectura típica de un SOM. Un SOM está compuesto por dos capas:

- La capa de entrada. El vector (una dimensión) $x(t)$ está formado por N neuronas ($x1$, $x2$, …, xN) que forman la primera capa.
- La capa de salida. La segunda capa es una matriz (dos dimensiones) encargada de proceder la información y formar el mapa.

Las conexiones entre las capas es unidireccional en sentido entrada -salida.

En los SOMs se utiliza el índice i,j para indicar la posición de la neurona y el índice k para indicar la conexión de la neurona de entrada. Por lo cual, los pesos sinápticos asociados a cada neurona tendrán tres índices.

Los pesos sinápticos son los valores que se les asocia a una neurona ya entrenada, aunque inicialmente se le asigna un vector de pesos aleatorio. Estos pesos indican la importancia que tiene la entrada en una neurona. El vector de pesos se representa con w_{ijk}.

3.2.3 ALGORITMO

Como se ha visto en el ***apartado 3.1.1 Fase de Entrenamiento*** las redes neuronales deben pasar por una fase de entrenamiento. El entrenamiento en los SOMs consiste en construir el mapa según unas muestras iniciales.

El algoritmo de aprendizaje de los mapas autoorganizados es el de Kohonen. A continuación, se describen los pasos. [11]

1. Iniciación de los pesos sinápticos w_{ijk}.
2. Elección de un patrón de entre el conjunto de patrones de entrenamiento.
3. Para cada neurona del mapa, calcular la distancia euclídea entre el patrón de entrada x y el vector de pesos sinápticos w_{ijk}.

$$d^2(\mathbf{w}_{ij}, \mathbf{x}) = \sum_k (w_{ijk} - x_k)^2$$

Ecuación 1. Distancia euclídea entre el vector sináptico y la entrada.

4. Evaluar la neurona ganadora (aquella cuya distancia es la menor de todas).
5. Actualizar los pesos sinápticos de la neurona ganadora y de sus vecinas según la regla de actualizaciones de pesos:

$$\delta w_{ijk}(t) = \alpha(t) \cdot h(|\mathbf{i} - \mathbf{g}|, t) \cdot (x_k(t) - w_{ijk}(t))$$

Ecuación 2. Actualización de los pesos en una red SOM.

$\alpha(t)$ es un factor llamado ritmo de aprendizaje que da cuenta de la importancia que la diferencia entre el patrón y los pesos tiene en el ajuste de los mismos a lo largo del proceso de aprendizaje.

$h\ (t)$ es la función de vecindad que indica en qué medida se modifican los pesos de las neuronas vecinas. Cuando la neurona ganadora modifica sus pesos, la vecindad de esta neurona lo hace también, en mayor o menor medida según sea la forma funcional de h. En general, las funciones empleadas para h tienen un máximo en $|i-j|=0$ y decrecen más o menos rápido a medida que esta distancia aumenta.

6. Lo usual es fijar un numero de iteraciones antes de comenzar el aprendizaje. Si no se llegó al número de iteraciones establecido previamente, se vuelve al paso 2. Sobre este número de iteraciones necesario, se suelen tomar criterios como el número de neuronas en el mapa.

Capítulo 4. PRIMER DISEÑO

En este capítulo se describen los pasos que se llevaron a cabo en la ***Etapa I. Búsqueda de Materiales para el Equipo*** y la ***Etapa II. Diseño y Construcción del sonómetro*** en un primer diseño.

El primer diseño consistió en hacer el equipo lo más económico posible. Este punto, llevó a utilizar un chip integrado con conexión wifi junto a un sonómetro.

A continuación, se explican las tecnologías y las conclusiones finales del primer diseño.

4.1 ESP8266

En un primer momento se decidió usar el ESP8266 para hacer la conexión inalámbrica. El ESP8266 es un chip integrado con conexión wifi, compatible con el protocolo TCP/IP.

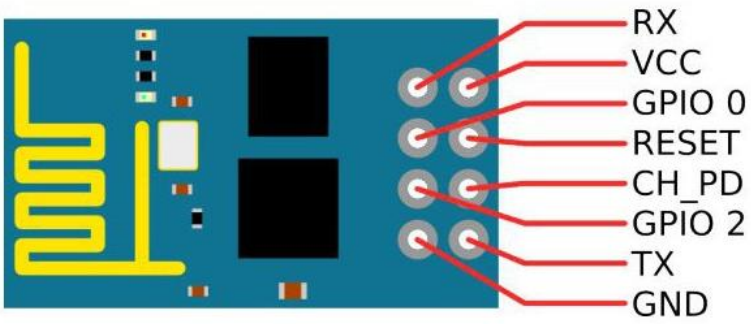


Figura 4. Esquema del ESP8266.

La ventaja de este chip es que además de los SDK (sistema operativo del módulo), se pueden volcar al chip firmwares alternativos que nos dan la opción de cargar un programa

directamente al módulo. En este caso, se usó el firmware NodeMCU. El precio de este chip ronda los 6€.

4.2 NODEMCU

NodeMCU es una plataforma de código abierto, que permite volcar en el ESP8266, el programa que se desee. El firmware se programa en lenguaje Lua. La principal ventaja que tiene usar un firmware como NodeMCU, es que te ahorras la placa *Arduino*, y por lo cual tienes un dispositivo IoT más económico y pequeño.

Para construir el firmware, simplemente se seleccionan los módulos deseados. En este caso, se han dejado los módulos por defecto y se han incluido el ADC (para lectura analógica) y el MQTT (protocolo).

4.3 PROTOCOLO MQTT

El protocolo MQTT es un protocolo de conectividad basado en M2M y orientado a IoT. Es un protocolo ideal para las conexiones en localizaciones remotas, donde no se tiene un código muy extenso y no se necesite seguridad extrema.

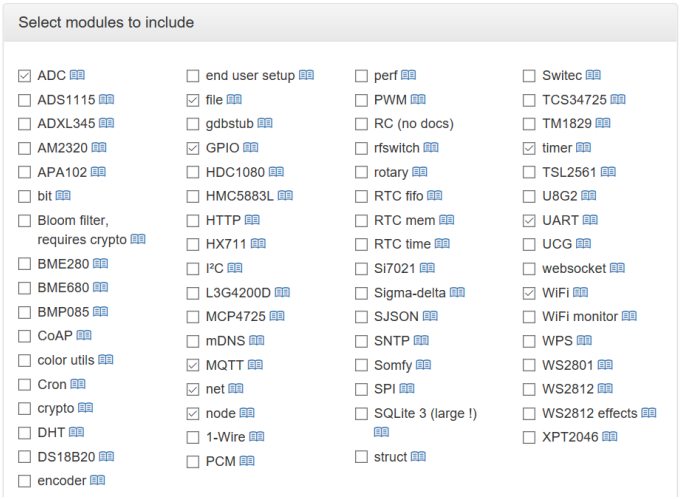


Figura 5. Módulos para construir el firmware.

Una vez volcado el programa en el chip, se puede acceder a él mediante el programa de desarrollo del ESPlorer.

4.4 SONÓMETRO MAX 9814

El sensor de sonido que se utiliza en el proyecto es el sensor MAX9814. Este sensor de sonido incluye un micrófono de 20 a 20kHz, lo que es ideal para un sensor como el que se necesita en este proyecto, que es simplemente un "reactivo de sonido".

Una de las ventajas de este componente es que elimina el ruido proveniente de la alimentación, lo que hace que el sonido que se obtiene es bastante preciso. [12]

Las características técnicas de este sensor son las siguientes:

- Por defecto el nivel máximo de ganancia es de 60dB, pero se puede cambiar puenteando los pines Gain, VCC y GND a 50dB o a 40 dB.
- El voltaje de alimentación es de 2,7 V a 5,5 V @3mA. Este voltaje es perfecto para conectarlo a una placa de *Arduino* ya que se hará la alimentación con el pin de 3,3 V de *Arduino*.
- La salida desde el amplificador es de 2Vpp max. sobre una base DC de 1,25V.
- Los pines son: AR, Out, Gain, Vdd y GND.

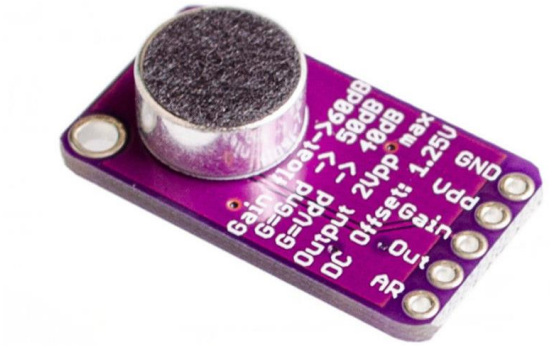


Figura 6. Sensor MAX9814.

Este sensor es el utilizado también en el diseño final y el precio ronda los 5€.

4.5 PROBLEMAS ENCONTRADOS

El problema que se encontró fue que el ESP8266 no tiene entrada analógica. La única solución que se puede dar a este problema es poner una de las patas del microchip a tierra. Esta solución no es muy adecuada ya que solo nos dio problemas. Las patas del microchip son milimétricas, y la conexión puede fallar.

Una vez hecha la conexión a tierra se programó el programa sin éxito, ya que daba problemas de memoria y de conexión. Por esta razón se decidió cambiar de diseño.

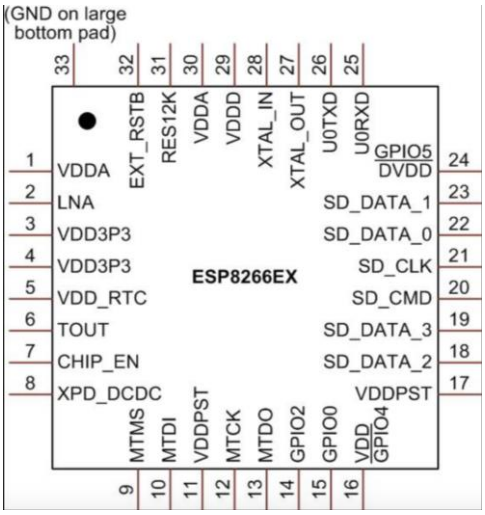


Figura 7. Microchip del ESO8266 con la pata 33 a GND.

Capítulo 5. DISEÑO FINAL

Finalmente, en este proyecto se ha optado por un método mucho más fácil de usar y que no da problemas y es el uso de una placa de *Arduino*.

En este capítulo se vuelven a repetir los pasos usados para las dos primeras etapas.

A continuación, se hace una explicación de las tecnologías que se han empleado en el proyecto.

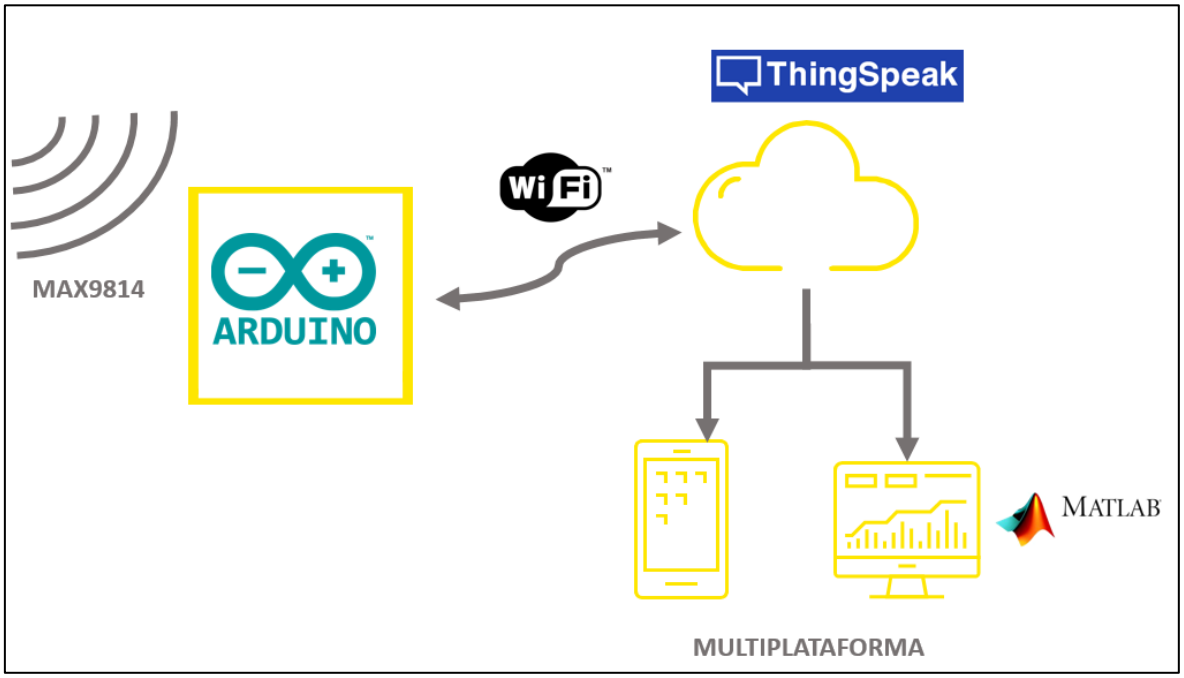


Figura 8. Esquema Alto Nivel.

5.1 ARDUINO

Arduino es una plataforma de creación electrónica de código abierto, con hardware y software libre, muy fácil de entender y utilizar. [13]

Las placas de *Arduino* están basadas en el microcontrolador ATmega328. Estas placas cuentan con entradas y salidas digitales y analógicas. Las ventajas que tiene usar una placa *Arduino* son:

- Flexibilidad. *Arduino* tiene hardware y software ampliable y de código abierto, lo que permite trabajar en todas las plataformas informáticas.
- Fácil de utilizar. No se necesita tener grandes conocimientos de programación para saber utilizar la placa. *Arduino IDE* es un entorno muy intuitivo, el cual se programa con lenguaje basado en C++.
- Plataforma *Arduino IDE*. Una de las ventajas de *Arduino*, es que ofrece un entorno de programación propio para utilizar sus placas. En el microcontrolador de *Arduino*, se escribe el lenguaje de programación para poder interactuar con los circuitos de la placa.
- Documentación. Gracias a que la marca *Arduino* es conocida mundialmente en el mundo del desarrollo, hay mucha documentación en internet y se obtiene mucha información fácilmente.

5.1.1 ARDUINO YUN

La placa *Arduino YUN* es una placa, perteneciente al grupo *Arduino*, que tiene conexión avanzada de redes y aplicaciones. La placa se conecta a la red wifi de forma sencilla gracias al panel web de *YUN*. Este panel, permite administrar la configuración de la placa y la carga de sketch. [14]

Figura 9. Placa Arduino YUN.

Esta placa de *Arduino* tiene conexión wifi al combinar el microcontrolador de *Arduino* con un módulo incorporado de tipo *SOC (System-On-a-Chip)*. Esta combinación, hace de la placa un componente muy potente en el proyecto al combinar la potencia de Linux junto con la sencillez que caracteriza a *Arduino*.

Cabe destacar que no solo soporta una red wifi, sino que también soporta red cableada Ethernet, pero en este proyecto no se utiliza al querer conexión inalámbrica por lo que no se hablará de ella en el documento.

Para usar la placa, es necesaria la librería *Bridge*. Esta librería tiene la función de hacer la conexión entre el microcontrolador y el módulo.

Las características más importantes de *Arduino YUN* son:

- Procesador: Atheros AR9331
- Arquitectura: MIPS @400MHz
- Alimentación: 3.3V
- Conexión wifi: IEEE 802.11b/g/n
- USB Type-A: 2.0 Host/Device

- Lector de tarjetas: Micro-SD
- RAM: 64 MB DDR2
- Memoria Flash: 32 MB
- Soporte para PoE tipo 802.3ª

5.1.2 ARDUINO IDE

Arduino IDE es el entorno de desarrollo en el que se programan las placas de *Arduino*. *Arduino IDE* tiene su propio lenguaje de programación, pero está basado en el lenguaje de programación C++. Los programas que se descargan en la placa se llaman *skecth*.

5.2 SONÓMETRO MAX 9814

El sensor de sonido que se utiliza en el proyecto es el sensor MAX9814. Es el mismo sensor que el utilizado en el diseño inicial, se encuentra detallado en el ***apartado 4.4.***

5.3 THINGSPEAK

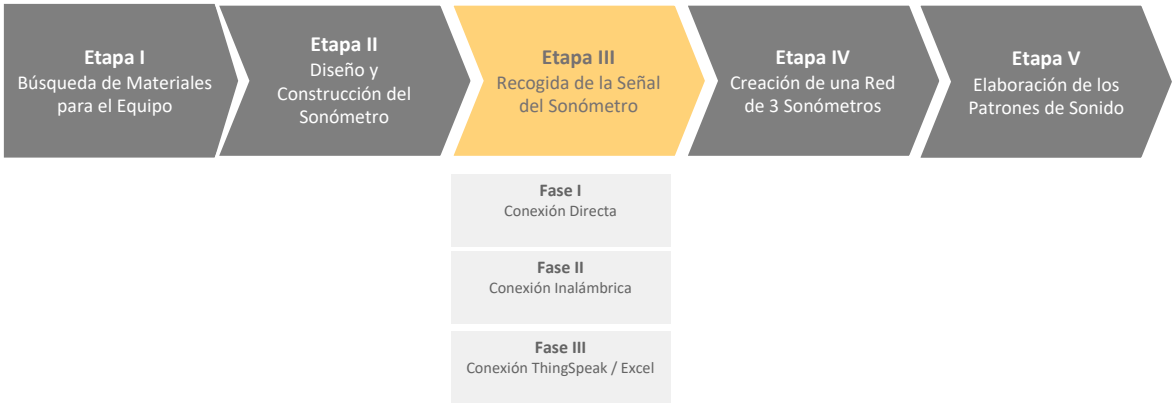
ThingSpeak es la plataforma web de base de datos enfocada al uso del IoT. La ventaja de *ThingSpeak* es que tiene compatibilidad con *Matlab*, la herramienta que se usará para desarrollar el mapa de sonido.

La ventaja de trabajar con *ThingSpeak* es que contiene librerías para *Arduino* (también contiene una librería para ESP8266).

Capítulo 6. RECOGIDA DE LA SEÑAL DEL SONÓMETRO

Las conexiones entre los diferentes componentes del sistema de audio no son complicadas, pero un mínimo fallo puede llevar a la destrucción total o parcial y con ello, a la inutilización del componente porque haya resultado dañado.

Las fases en las que se compone la implementación son las siguientes:



6.1.1 FASE I CONEXIÓN DIRECTA

La primera fase es **comprobar el funcionamiento del sonómetro**. Para esta fase no se ha utilizado la conexión wifi, se hace conexión directa con el mini USB de la placa.

En primer lugar, se conecta el sonómetro con la placa de *Arduino*. La conexión sería la siguiente:

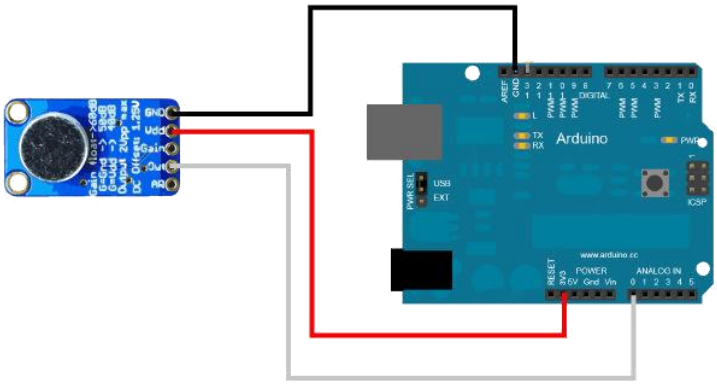


Figura 10. Conexión Arduino con el MAX9814.

Primeramente, se programa el código para captar el sonido del sonómetro. El código que se usa en esta fase es el que proporciona el fabricante de MAX9814, con algunas modificaciones que se han añadido. Es un código sencillo, pero en esta etapa es suficiente. Más adelante se completará el código con los parámetros necesarios. El código es el siguiente:

```
const int sampleWindow = 250; // Sample window width in mS (250 mS = 20Hz)
unsigned int sample;

void setup()
{
   Serial.begin(9600);
}

void loop()
{
  unsigned long startMillis= millis();  // Start of sample window
  unsigned int peakToPeak = 0;   // peak-to-peak level

   unsigned int signalMax = 0;
   unsigned int signalMin = 1024;

   // collect data for 250 mS
     while (millis() - startMillis < sampleWindow)
     {
        sample = analogRead(0);
        if (sample < 1024)  // toss out spurious readings
        {
           if (sample > signalMax)
           {
```

```
            signalMax = sample;  // save just the max levels
         }
         else if (sample < signalMin)
         {
            signalMin = sample;  // save just the min levels
         }
      }
   }
   peakToPeak = signalMax - signalMin;  // max - min = peak-peak amplitude
   double volts = (peakToPeak * 10.0) / 1024;  // convert to volts

   if(volts<0.5){
    Serial.print(" Volts: " + String(volts));
    Serial.print(" Def: susurro");
    Serial.println("");
   }else if(volts<1.5){
    Serial.print(" Volts: " + String(volts));
    Serial.print(" Def: bajo");
    Serial.println("");
   }else if(volts < 2){
    Serial.print(" Volts: " + String(volts));
    Serial.print(" Def: medio");
    Serial.println("");
   }else if(volts < 2.5){
    Serial.print(" Volts: " + String(volts));
    Serial.print(" Def: alto");
    Serial.println("");
   }else if(volts < 3){
    Serial.print(" Volts: " + String(volts));
    Serial.print(" Def: muy alto");
    Serial.println("");
   }
}
```

Lo que hace este código escoger es muestras de sonido cada 250 m segundos y convertirlas a voltios.

La salida del *sketch* es la siguiente:

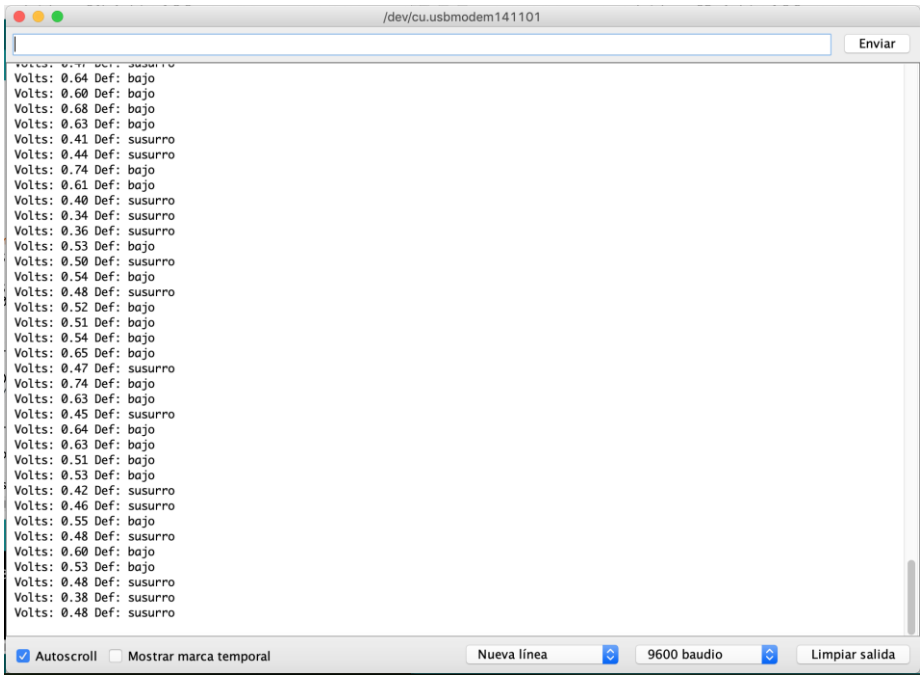


Figura 11. Salida del sketch.

Una vez comprobado el funcionamiento del sonómetro, se comprueba la comunicación inalámbrica.

6.1.2 FASE II CONEXIÓN INALÁMBRICA

El objetivo de la segunda fase es **comprobar la conexión inalámbrica** por medio de SSH. SSH es un protocolo muy fácil de usar y *Arduino YUN* es capaz de establecer esta comunicación a través de SSH con un equipo. Para hacer una comunicación SSH, se ha utilizado un software gratuito llamado PuTTY. Este software solo trabaja en Windows, y funciona simplemente como terminal.

Una vez que el código está subido a la placa de *Arduino YUN* (primera fase) se configura la comunicación wifi.

Los pasos son los siguientes:

1. Se resetea el wifi del *Arduino YUN*. Hay que mencionar que *Arduino YUN* funciona muy parecido a un *router* de wifi, por lo que cuando conectemos el wifi, no

tendremos conexión de inmediato. Se presiona el botón de *reset* durante 30 segundos, esto provocará que se reinicien las configuraciones por defecto.

2. Cuando se resetea, aparecerá una conexión disponible desde nuestro ordenador a este wifi. Se conecta a dicha red.
3. Se accede a un navegador y tecleamos "arduino.local".
4. Se introduce la contraseña de *Arduino YUN*. Por defecto es "arduino".
5. Se accede a una página de configuración, en la que podremos cambiar la contraseña, el nombre, añadir la red wifi… En esta página se puede consultar datos como la dirección IP y la dirección MAC. En este caso, al ser la primera placa, se configura como "**arduino1**".
6. Cuando se termina la configuración, la placa está lista para utilizarse por medio wifi.

Una vez configurado el wifi, se comprueba la comunicación. Para este paso, se trabaja con PuTTY.

Antes de trabajar con PuTTY, se debe hacer un cambio en el código. La placa "debe saber" cuando tiene que conectarse y enviar la información. El código, por lo tanto, será el siguiente:

```
#include <Console.h>

const int sampleWindow = 250; // Sample window width in mS (250 mS = 20Hz) 20
muestras por segundo
unsigned int sample;

void setup()
{
  // Se inicia la comunicación
  Bridge.begin();
  Console.begin();

  while (!Console){
    ; // Se espera hasta que se conecte la consola
  }
  Console.println("Se ha conectado a la consola!!!!");
  Serial.begin(9600);
}

void loop()
  {
```

```
    if (Console.available() > 0) {
    unsigned long startMillis= millis();  // Start of sample window
    unsigned int peakToPeak = 0;   // peak-to-peak level

     unsigned int signalMax = 0;
     unsigned int signalMin = 1024;

     // collect data for 250 mS
       while (millis() - startMillis < sampleWindow)
       {
          sample = analogRead(0);
          if (sample < 1024)  // toss out spurious readings
          {
             if (sample > signalMax)
             {
                signalMax = sample;  // save just the max levels
             }
             else if (sample < signalMin)
             {
                signalMin = sample;  // save just the min levels
             }
          }
       }
       peakToPeak = signalMax - signalMin;  // max - min = peak-peak amplitude
       double volts = (peakToPeak * 10.0) / 1024;  // convert to volts

       if(volts<0.5){
        Console.print(" Volts: " + String(volts));
        Console.print(" Def: susurro");
        Console.println("");
       }else if(volts<1.5){
        Console.print(" Volts: " + String(volts));
        Console.print(" Def: bajo");
        Console.println("");
       }else if(volts < 2){
        Console.print(" Volts: " + String(volts));
        Console.print(" Def: medio");
        Console.println("");
       }else if(volts < 2.5){
        Console.print(" Volts: " + String(volts));
        Console.print(" Def: alto");
        Serial.println("");
       }else if(volts < 3){
        Console.print(" Volts: " + String(volts));
        Console.print(" Def: muy alto");
        Console.println("");
       }
    }
    delay(100);
}
```

Se accede a la configuración de la placa de *Arduino YUN*, para comprobar la dirección IP de la placa.

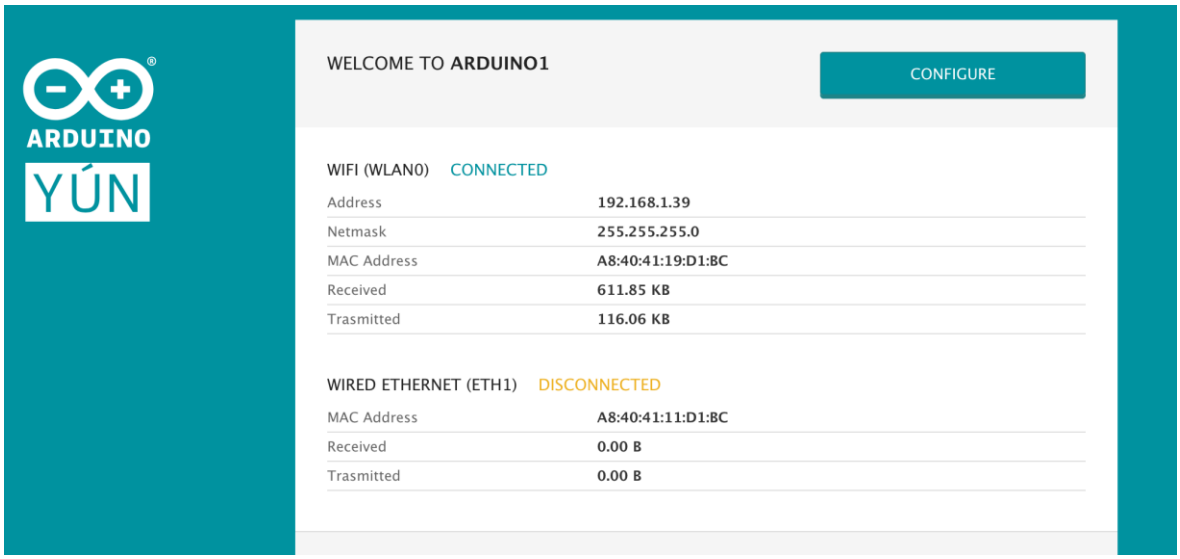


Figura 12. Página de configuración del Arduino desde el navegador.

A continuación, se abre el PuTTY y se configura la comunicación. Se añade la IP de la placa y ya está listo para funcionar.

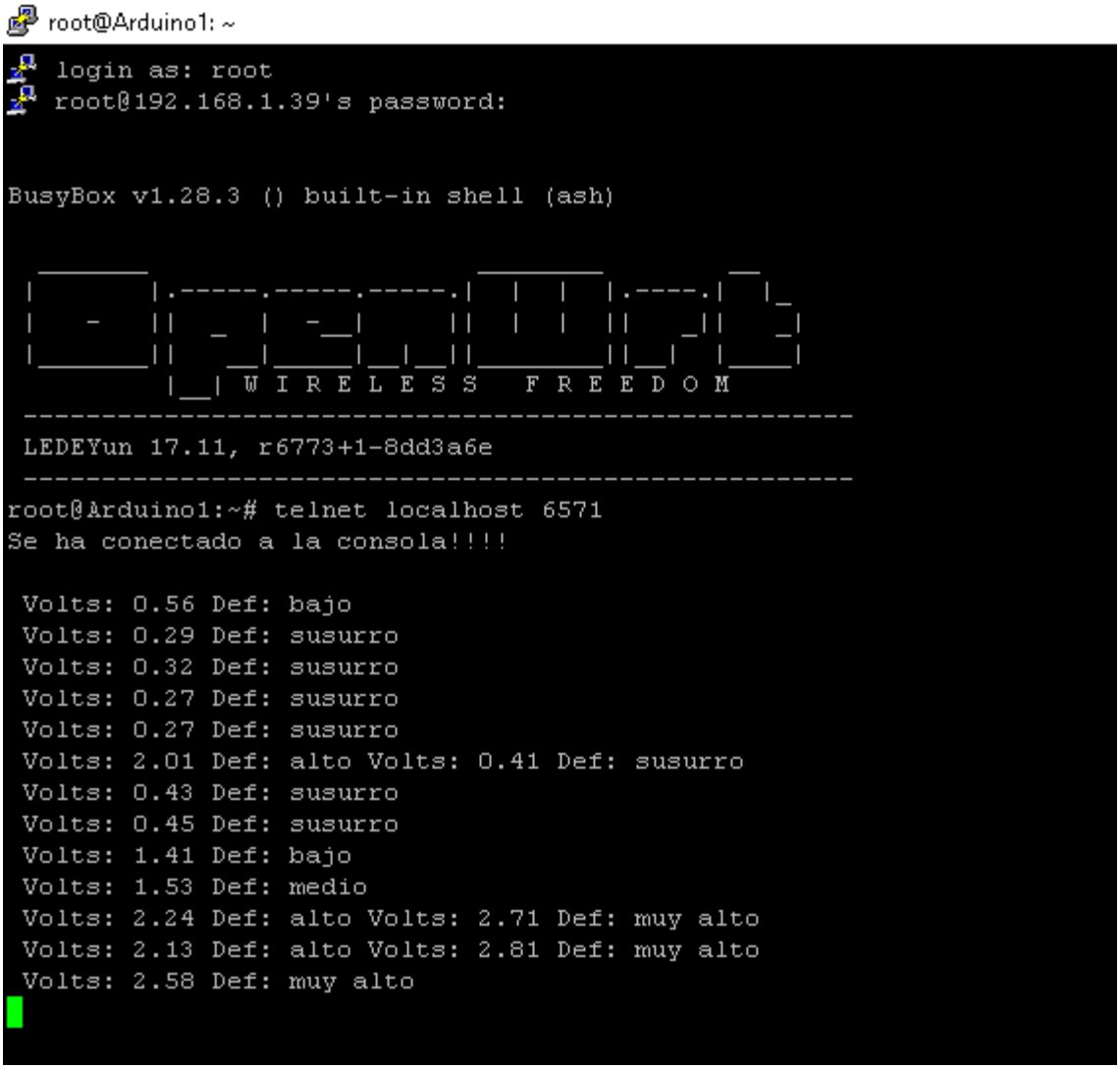


Figura 13. Consola del Putty.

6.1.3 FASE III CONEXIÓN THINGSPEAK / EXCEL

Una vez comprobado que la conexión inalámbrica de la placa funciona, se **comprueba el funcionamiento con *ThingSpeak* y con Putty-Excel.**

Se decide probar las dos plataformas para discutir cuál es más cómoda a la hora de trabajar con *Matlab*.

6.1.3.1 ThingSpeak

Para empezar la conexión con *ThingSpeak*, se crea un canal en la plataforma web. El canal creado tiene como ID: "785490" y como clave para escribir en *ThingSpeak*: "Q5K2TQROZOO5RGU6".

El código se modifica de la siguiente manera:

```
#include "ThingSpeak.h"
#include <BridgeClient.h>

BridgeClient client;
unsigned long myChannelNumber = 785490;
const char * myWriteAPIKey = "Q5K2TQROZOO5RGU6";

const int sampleWindow = 250; // Sample window width in mS (250 mS = 20Hz) 20
muestras por segundo
unsigned int sample;

void setup() {
  Serial.begin(9600);  //Initialize serial
  Serial.println("Empieza");
   while (!Serial) {
   ; // wait for serial port to connect. Needed for native USB port only
    }

  Bridge.begin();
  ThingSpeak.begin(client);  // Initialize ThingSpeak
}

void loop() {

  // Write to ThingSpeak. There are up to 8 fields in a channel, allowing you to
store up to 8 different
  // pieces of information in a channel.  Here, we write to field 1.


     unsigned long startMillis= millis();  // Start of sample window
     unsigned int peakToPeak = 0;   // peak-to-peak level

     unsigned int signalMax = 0;
     unsigned int signalMin = 1024;

     // collect data for 250 mS
     while (millis() - startMillis < sampleWindow)
     {
        sample = analogRead(0);
        if (sample < 1024)  // toss out spurious readings
        {
           if (sample > signalMax)
```

```
            {
               signalMax = sample;  // save just the max levels
            }
            else if (sample < signalMin)
            {
               signalMin = sample;  // save just the min levels
            }
         }
      }
      peakToPeak = signalMax - signalMin;  // max - min = peak-peak amplitude
      float volts = (peakToPeak * 10.0) / 1024;  // convert to volts
      int x = ThingSpeak.writeField(myChannelNumber, 1, volts, myWriteAPIKey);
      Serial.println(volts);
      delay(10000); //10 segundos
}
```

Los resultados del *Arduino* son cada 10 segundos (de ahí viene el `delay(10000)`) en cambio, en la plataforma de *ThingSpeak*, los resultados llegan cada 20 segundos (esta plataforma no puede enviar más rápido que 15 segundos). Por lo que los resultados no son tan exactos. A continuación, se muestran las salidas de las dos plataformas, y se comprueba que el *Arduino IDE* tiene el doble de los resultados.

Cabe destacar, que los valores y los tiempos en varias plataformas son exactos, a pesar de que en una haya el doble de valores.

```
20:01:42.838 -> 0.21
20:01:53.665 -> 0.25
20:02:04.584 -> 0.21
20:02:15.434 -> 0.30
20:02:26.251 -> 2.02
20:02:37.016 -> 3.07
20:02:47.863 -> 2.81
20:02:58.694 -> 1.97
20:03:09.780 -> 2.60
20:03:20.601 -> 3.12
20:03:31.405 -> 3.15
20:03:42.278 -> 2.80
20:03:53.109 -> 1.66
20:04:04.032 -> 1.70
20:04:15.259 -> 2.83
20:04:26.054 -> 3.23
20:04:36.877 -> 2.19
20:04:47.676 -> 2.71
20:04:58.524 -> 2.83
20:05:09.399 -> 3.05
20:05:20.598 -> 3.24
20:05:32.258 -> 3.04
```

Figura 14. Salida del Arduino IDE (Voltios).

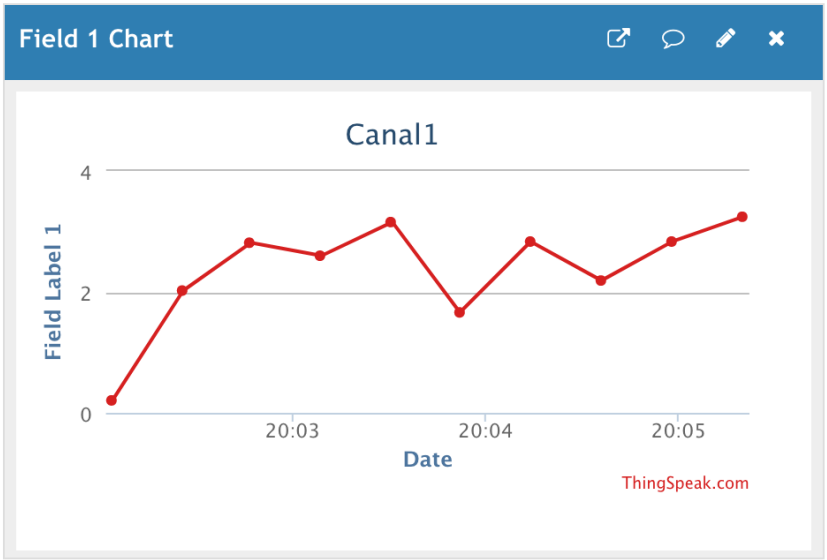


Figura 15. Salida del ThingSpeak (Voltios).

6.1.3.2 Putty-Excel

Otra forma de trabajar los datos con *Matlab* es extrayendo los datos desde Excel. Hay una manera muy cómoda y sencilla que es volcar los datos recibidos de Putty a una hoja de Excel.

La utilización de Putty se explica en el ***apartado 6.1.2. Fase II.***

Para volcar los datos a una hoja de Excel, simplemente hay que seleccionar la opción de "Printable output" y elegir donde guardar el documento (indicando la extensión).

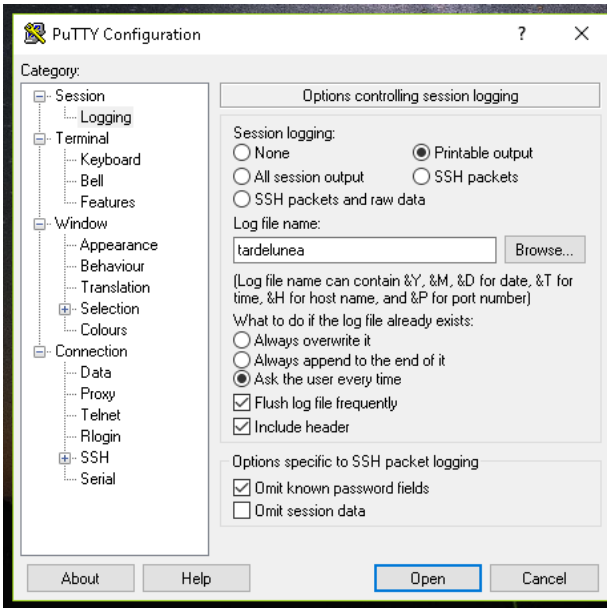
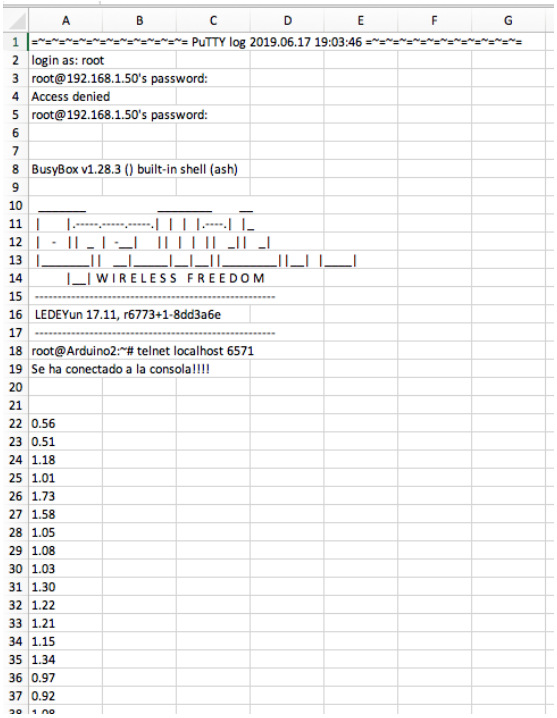


Figura 16. Configuración Putty.



	A	B	C	D	E	F	G
1	=~=~=~=~=~=~=~=~=~=~= PuTTY log 2019.06.17 19:03:46 =~=~=~=~=~=~=~=~=~=~=						
2	login as: root						
3	root@192.168.1.50's password:						
4	Access denied						
5	root@192.168.1.50's password:						
6							
7							
8	BusyBox v1.28.3 () built-in shell (ash)						
9							
10	_____ _____ __						
11	| | .-----.-----.-----.| | | |.----.| |_						
12	| - || _ | -__| || | | || _|| _|						
13	|_____|| __|_____|__|__||_____||__| |____|						
14	|__| W I R E L E S S F R E E D O M						
15	-----------						
16	LEDEYun 17.11, r6773+1-8dd3a6e						
17	-----------						
18	root@Arduino2:~# telnet localhost 6571						
19	Se ha conectado a la consola!!!!						
20							
21							
22	0.56						
23	0.51						
24	1.18						
25	1.01						
26	1.73						
27	1.58						
28	1.05						
29	1.08						
30	1.03						
31	1.30						
32	1.22						
33	1.21						
34	1.15						
35	1.34						
36	0.97						
37	0.92						

Figura 17. Ejemplo hoja Excel.

El código de programación del *Arduino* se modifica de la siguiente manera:

```
#include <Console.h>

const int sampleWindow = 250; // Sample window width in mS (250 mS = 20Hz) 20
muestras por segundo
unsigned int sample;

void setup()
{
// Se inicia la comunicación
Bridge.begin();
Console.begin();

while (!Console){
  ; // Se espera hasta que se conecte la consola
}
Console.println("Se ha conectado a la consola!!!!");
Serial.begin(9600);
}

void loop(){
  if (Console.available() > 0) {
  unsigned long startMillis= millis();  // Start of sample window
  unsigned int peakToPeak = 0;   // peak-to-peak level
```

```
   unsigned int signalMax = 0;
   unsigned int signalMin = 1024;

   // collect data for 250 mS
     while (millis() - startMillis < sampleWindow)
     {
        sample = analogRead(0);
        if (sample < 1024)  // toss out spurious readings
        {
           if (sample > signalMax)
           {
              signalMax = sample;  // save just the max levels
           }
           else if (sample < signalMin)
           {
              signalMin = sample;  // save just the min levels
           }
        }
     }
     peakToPeak = signalMax - signalMin;  // max - min = peak-peak amplitude
     double volts = (peakToPeak * 10.0) / 1024;  // convert to volts

     Console.print(String(volts));
     Console.println("");
  }
  delay(100);
}
```

6.1.3.3 Conclusión

Si se necesitara mandar los datos en *Streaming*, se usaría *ThingSpeak*, pero en este proyecto no es necesario.

La forma más sencilla de mandar datos al código de *Matlab*, es mandando los datos a través de Excel, por lo que se trabajará de esta manera.

Capítulo 7. CREACIÓN DE LA RED DE LOS SONÓMETROS

Una vez que el sonómetro envía información de modo inalámbrico, se pasa a montar una red. Esta fase es sencilla ya que ya que son 3 sonómetros igualmente configurados y sin necesidad de sincronía.

El único requisito del proyecto es reconocer qué sonómetro es el que manda información.

Como finalmente se ha elegido el método de Putty-Excel, cada sonómetro enviará los datos de modo inalámbrico al servidor al mismo momento hasta recoger 100 muestras.

7.1 DESCRIPCIÓN DEL ENTORNO DE LA RED

Se decide colocar los 3 sonómetros en torno a un colegio. Un colegio puede ser un entorno interesante por la variedad de situaciones que pueden darse. La razón por la que se decide elegir dicho entorno es porque hay momentos de mucho ruido, como por ejemplo a la salida de clase o en el recreo, y otras de calma, por ejemplo, por la noche.

Cada sonómetro se ha colocado aproximadamente unos 20 metros uno del otro

Capítulo 8. ELABORACIÓN DE LOS PATRONES DE SONIDO

La última etapa de este proyecto es la elaboración de los mapas autoorganizados. Para ello, hay que recoger una gran cantidad de datos de sonido y pasar los mapas por la fase de entrenamiento y verificación para sacar conclusiones finales.

8.1 FASE I RECOGIDA DE DATOS

En esta primera fase, se recogen los datos con los que se van a formar los mapas.

La recogida de datos se ha hecho siguiendo el siguiente esquema:

- Se distinguen 4 franjas horarias:
  - **08:00**: Primera hora de la mañana.
  - **12:00:** Medio día.
  - **18:00:** Tarde.
  - **00:00:** Madrugada.
- Se distingue si es **fin de semana, sábado o domingo.**
- Por cada situación, se recogen **100 muestras** de datos.

Con esta información, cada día se obtienen 400 muestras por sonómetro, por lo cual:

$$400\ muestras\ /\ sonómetro\ \times\ 3\ sonómetros = 1.200\ muestras\ al\ día$$

En total para este proyecto se habrán recogido:

$$1.200\ \text{muestras}\ /\ \text{día}\ \times\ 7\ \text{días}\ =\ \mathbf{8.400\ muestras\ en\ total}$$

Cada muestra tendrá los siguientes **atributos**:

Franja horaria	8:00	12:00	18:00	0:00
Valor	0001	0010	0100	1000

Tabla 2. Valores del atributo franja horaria.

Día de la semana	Lunes a Viernes	Sábado	Domingo
Valor	001	010	100

Tabla 3. Valores del atributo día de la semana.

El motivo por el que se han puesto valores binarios a los atributos es para que no tengan dependencia temporal. Esta razón se explica en el ***siguiente apartado***.

8.2 FASE II ENTRENAMIENTO

Para la fase de entrenamiento se han cogido 70 muestras al azar de las 100 que se obtiene en cada franja horaria. Es importante que sea al azar para poder romper la dependencia temporal.

Las entradas para crear el SOM serán:

s1	**s2**	**s3**	**fh**	**d**
Muestra Sonómetro 1	Muestra sonómetro 2	Muestra sonómetro 3	Franja horaria	Día de la semana
Valor (0-4)	Valor (0-4)	Valor (0-4)	Vector [4]	Vector [3]

Tabla 4. Entradas del SOM.

Como se ha explicado en el apartado anterior, para que no haya dependencia temporal (tanto en la franja horaria como en el día de la semana) se han utilizado valores binarios y se colocan distintas entradas. Por tanto, las entradas (o atributos) serán:

1	2	3	4	5	6	7	8	9	10
s1	s2	s3	08:00	12:00	18:00	00:00	L-V	S	D
ID del sonómetro			Franja horaria				Día de la semana		

Tabla 5. Atributos del SOM.

La forma más sencilla para crear un SOM con *Matlab* es coger los datos desde una matriz. Esta matriz estará en Excel y tendrá recogidos todos los datos recogidos de esa semana.

La forma de crear la matriz es poniendo por columnas las muestras de cada sonómetro y su franja horaria y el día de la semana en forma binaria, como se ha explicado anteriormente.

La estructura de la matriz es la siguiente:

Muestra	s1	s2	s3	fh	d
1	DATOS	DATOS	DATOS	DATOS	DATOS
2	DATOS	DATOS	DATOS	DATOS	DATOS
…	DATOS	DATOS	DATOS	DATOS	DATOS

Tabla 6. Estructura de la Matriz de datos (Macrotabla).

Una vez que se tiene la *Macrotabla* (así es como se le llamará a la Matriz de datos), se cogerán 70 muestras aleatorias por franja, para poder empezar la fase de entrenamiento.

$$70\ muestras\ \times\ 4\ franjas\ horarias\ \times\ 3\ tipos\ de\ día = 840\ muestras$$

$$840\ muestras\ \times\ 3\ sonómetros = 2.520\ datos\ de\ los\ sonómetros$$

El resultado será una nueva matriz de 5 columnas por 840 filas. A esta matriz se le llamará *Matriz de Entrenamiento.*

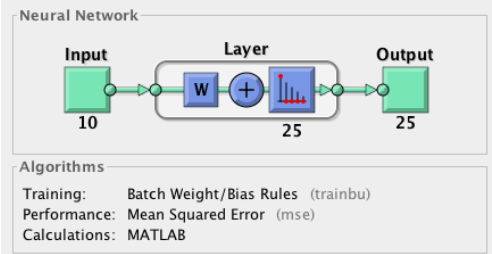


Figura 18. Arquitectura SOM del proyecto.

En la ***Figura anterior*** se puede ver la forma de funcionar de esta red. La explicación es la siguiente:

- Los datos de entrada (10 atributos) son enviados a la capa de neuronas. En este proyecto solo hay una capa de neuronas y se compone de 25 neuronas (dimensión 5 x 5).
- Finalmente, la salida del proceso son 25 patrones.

El código fuente usado para el SOM se encuentra en el ***Anexo I***.

8.3 RESULTADOS ENTRENAMIENTO

A continuación, se van a analizar los mapas y gráficas que se han obtenido en la fase de entrenamiento.

En primer lugar, se representan los datos recogidos. Se observa que por cada sonómetro se han recogido 840 muestras. El eje Y representa los Voltios y el eje X las muestras.

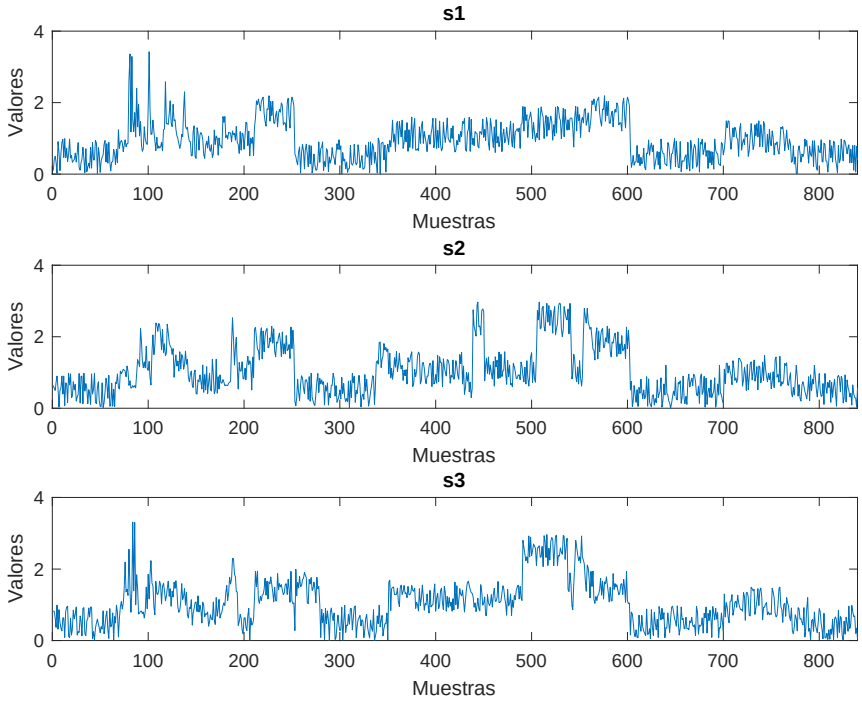


Figura 19. Representación datos Matriz de Entrenamiento.

A primera vista se puede observar que de la muestra 0 a la 281, que son las muestras que corresponden a los días laborables, hay mucha variedad de valores.

Los valores de la muestra 281 a la 561, que son las muestras correspondientes al sábado, son más altos que los días laborables.

Y por último, las últimas muestras de la 561 a la 840, representan el domingo y tienen valores mucho más bajos.

Estas conclusiones se analizan con más detalle en los siguientes apartados.

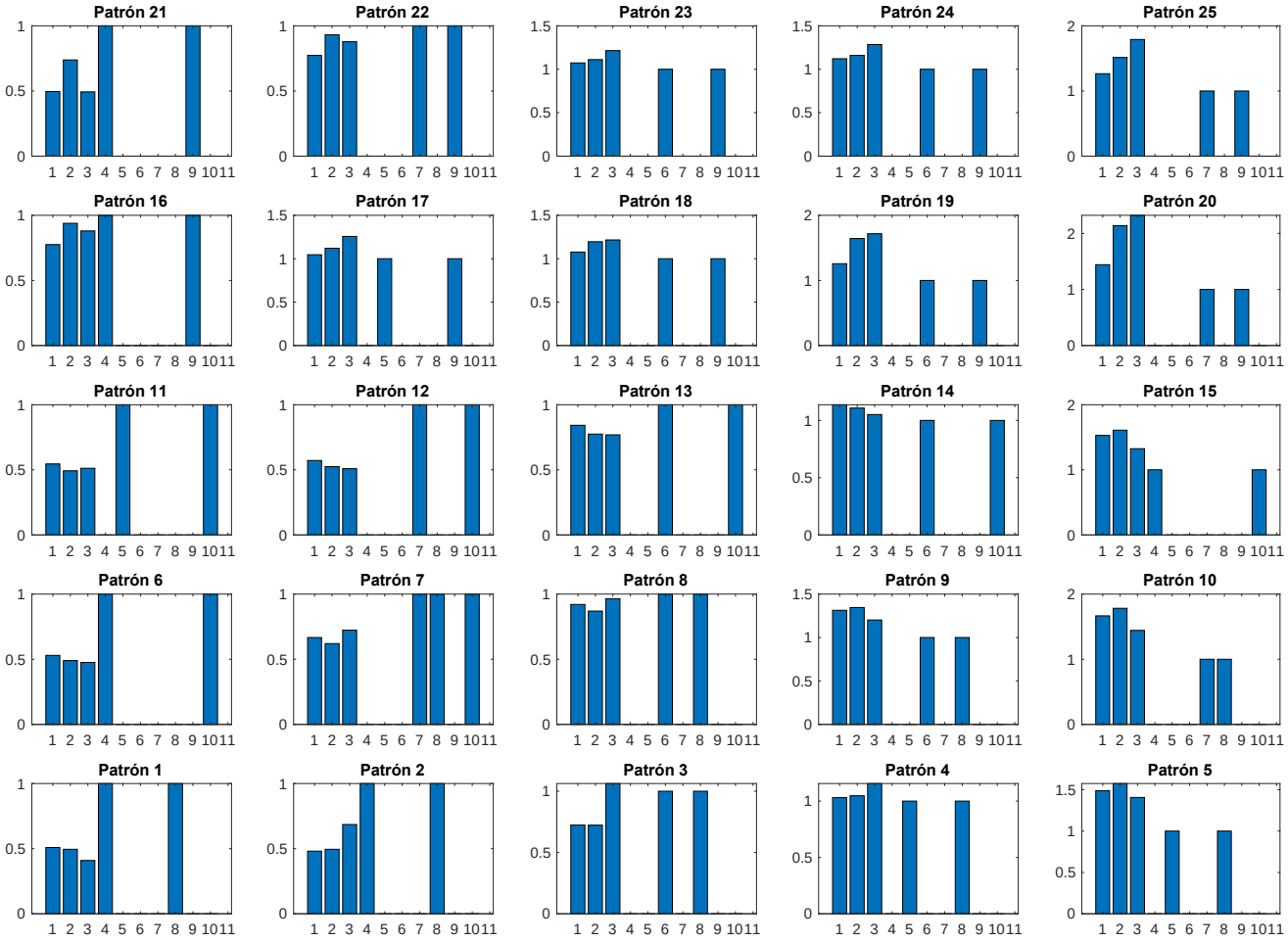


Figura 20. Patrones

Una primera comprobación para ver si la red está funcionando bien es verificar que los patrones se hayan ordenado correctamente según los atributos. En la ***Figura 20*** se puede ver cómo están ordenados:

- Las dos primeras filas de patrones corresponden al sábado (patrón 25 - patrón 20).
- La tercera fila y el patrón 6, corresponden a un día laborable.
- La última fila y los patrones 7, 8, 9 y 10 corresponden al domingo.

A continuación, se exponen las conclusiones de los patrones por día de la semana.

Lunes a Viernes

- Este tipo de día es el que menos patrones tiene porque es el que menos cambia. Tiene sentido ya que, de lunes a viernes, la gente sigue una rutina y así, el ruido que pueda producir.
- La franja horaria que tiene más ruido es sin duda la de las 12:00.
- Por lo general los sonómetros dan resultados parecidos.

Sábados

- Por lo general el sonómetro que da valores más bajos es el 1, después el 2 y por último, el que da valores más altos, es el sonómetro 3. Esto puede ser debido a su posición en el entorno.
- En cambio, si es a las 00:00, el sonómetro que da valores más altos es el 2.
- Los valores de los sonómetros a las 00:00 son los más bajos y los más altos son a las 8:00 y a las 12:00.

Domingos

- A las 12:00 a veces da el sonómetro 3 valores más altos y otras más bajos.
- A las 18:00 a veces salen valores más altos y otras muy bajos.

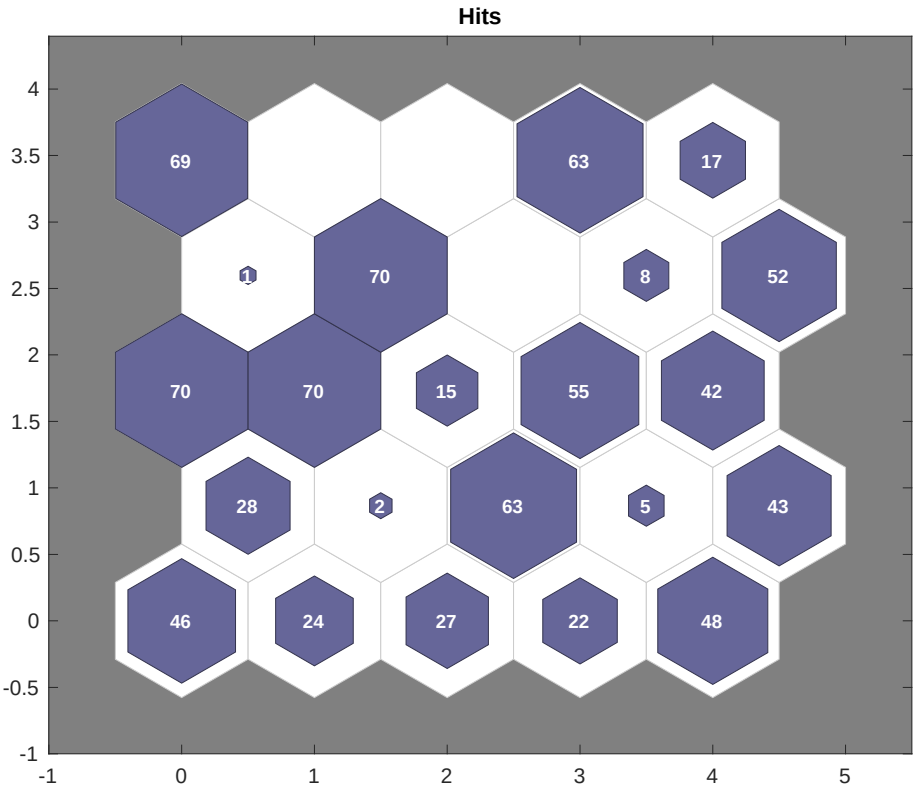


Figura 21. SOM Sample Hits

La ***figura anterior*** representa la red. Cada una de las formas hexagonales son las representaciones de las neuronas y el valor que tiene escrito dentro es la cantidad de muestras que caen en cada neurona.

Por ejemplo, para el caso de los sábados (las dos primeras filas de neuronas) hay un total de 281 muestras. Es correcto ya que deberían de haber caído 280 muestras.

Además, esta figura representa qué patrones son más comunes. Los patrones más comunes son:

- Patrón 21: Sábado a las 00:00. Los valores que dan los sonómetros no son excesivamente altos y el sonómetro 2 es el que da valores más altos.
- Patrón 17: Sábado a las 18:00. Los valores de los sonómetros son altos y el sonómetro que da valores más altos es el 3.

- Patrón 11: Día laborable a las 18:00. El valor de los sonómetros es bajo y los tres sonómetros dan valores parecidos.
- Patrón 12: Día laborable a las 08:00. El valor de los sonómetros es bastante parecido, aunque el sonómetro 1 da valores más altos, y son también valores bajos.

Los patrones en los que no ha caído ninguna muestra son:

- Patrón 22: Sábado a las 08:00. Valores altos de los sonómetros y el sonómetro 2 es el que da valores más altos.
- Patrón 23: Sábado a las 12:00. Valores altos y el sonómetro 3 es el que da los valores más altos.
- Patrón 18: Sábado a las 12:00. Valores altos y el sonómetro 3 es el que da los valores más altos (junto con el sonómetro 2).

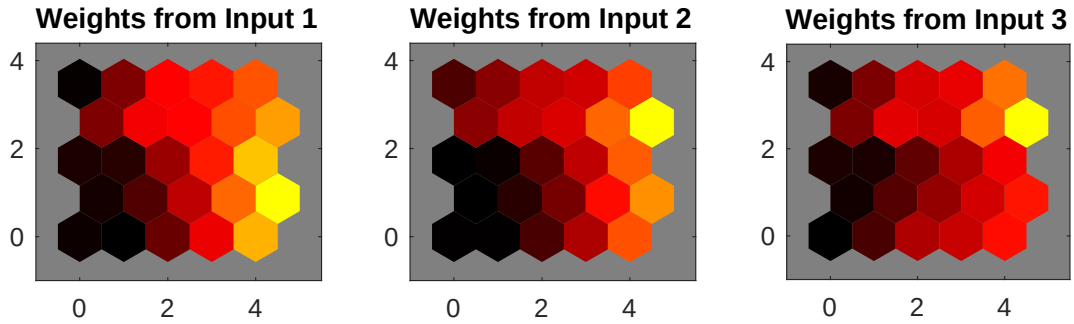


Figura 22. SOM Input Planes

La figura anterior representa la influencia de cada sensor, como entrada al mapa, en los patrones o pesos obtenidos tras el entrenamiento el mapa. Si una neurona participa mucho en el mapa es porque tiene valores parecidos con sus vecinas.

Si el color es muy oscuro, es porque su influencia no destaca (o es casi nula) en el mapa, y si el color tiende a rojo su contribución es muy importante. Esto refleja cómo los patrones obtenidos en la parte baja de la izquierda de las figuras, el nivel del sonido es mucho menor que en los patrones de la parte centro-derecha del mapa.

Los patrones de la parte central indican que los sonidos son los más importantes para explicar los patrones.

Los patrones de la parte derecha muestran un nivel sonoro inferior a los centrales siendo el sonómetro 1 el que capta menos intensidad en esos patrones y el que más, el sonómetro 3.

8.4 FASE III VERIFICACIÓN

Una vez que se tiene creada la red, se pasa a la fase de verificación. Esta fase es muy sencilla; se cogen las 30 muestras restantes que no se usaron en la fase de entrenamiento y se meten en el mapa a ver en qué neurona caen.

La modificación del código sería la siguiente:

```
%% Fase validación
prediccion= sim(net, input');
```

El código fuente completo se encuentra en el ***Anexo II***.

8.5 RESULTADOS VERIFICACIÓN

Los resultados de la fase de verificación se ven reflejados en la ***Tabla 7. Comparación entre la fase de entrenamiento y fase de verificación.***

																									TOTAL
Nº Patrón	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	
Fase Verificación	21	9	15	30	0	30	15	29	0	1	30	30	2	28	0	2	30	0	0	18	28	0	0	30	12	**360**
%	6%	3%	4%	8%	0%	8%	4%	8%	0%	0%	8%	8%	1%	8%	0%	1%	8%	0%	0%	5%	8%	0%	0%	8%	3%	100%
Fase Entrenamiento	46	24	27	22	48	28	2	63	5	43	70	70	15	55	42	1	70	0	8	52	69	0	0	63	17	**840**
%	5%	3%	3%	3%	6%	3%	0%	8%	1%	5%	8%	8%	2%	7%	5%	0%	8%	0%	1%	6%	8%	0%	0%	8%	2%	100%
Diferencia	0%	0%	1%	6%	-6%	5%	4%	1%	-1%	-5%	0%	0%	-1%	1%	-5%	0%	0%	0%	-1%	-1%	0%	0%	0%	1%	1%	

Tabla 7. Comparación entre la fase de entrenamiento y fase de verificación.

Capítulo 9. CONCLUSIONES Y TRABAJOS FUTUROS

Una vez que se han comparado las fases de entrenamiento y verificación y viendo que los porcentajes de error son muy bajos, se da por válida la red. Ningún patrón tiene un margen de error de más del 6%.

El objetivo del proyecto era realizar por un lado el equipo de sonido para poder recoger los datos y, por otro lado, analizarlos.

Gracias a los patrones recogidos, se puede ayudar a tomar decisiones sobre un posible impacto tanto en actividades de un entorno concreto, como en la planificación de otras en periodos donde se prevé un menor impacto sonoro exterior.

Sería interesante hacer este proyecto en *Streaming* ya que los datos que hemos recogido han sido *off-line* para analizarlos. Fácilmente se podría hacer con la herramienta de *ThingSpeak*, que se ha hablado durante el proyecto, ya que manda los datos directamente a Matlab. Además, para trabajos futuros, se podría desarrollar una aplicación móvil para recoger los datos más fácilmente. Esta idea se pensó al principio, pero por problemas de tiempo, no se llevó adelante.

Existen numerosos proyectos relacionados con SOMs pero pocos con mapas de sonido enfocados a un entorno en concreto, y como se explica en el ***apartado 1.2 Estado del arte***, hay muchas aplicaciones y se obtienen resultados muy precisos e interesantes.

Además, al ser un equipo de bajo coste y fácilmente configurable, se pueden añadir al equipo tantos sonómetros como se requiera y así hacer mapas aún más grandes.

Capítulo 10. BIBLIOGRAFÍA

[1] https://es.wikipedia.org/wiki/Internet_de_las_cosas

[2] https://es.wikipedia.org/wiki/MATLAB

[3] https://ra2017cnrs.fr/en/une-cartographie-collaborative-de-lenvironnement-sonore/

[4] Music Mood Visualization Using Self-Organizing Maps – Magdalena Plewa

[5] Application of Self-Organizing Maps to the Maritime Environment – Victor JAS Lobo

[6] Context-dependent environmental sound monitoring using SOM coupled with LEGION – Damiano Oldoni

[7] https://es.wikipedia.org/wiki/Red_neuronal_artificial

[8] https://www.youtube.com/watch?v=6vwfT3-mBBw

[9] https://es.wikipedia.org/wiki/Mapa_autoorganizado

[10] http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema5dm.pdf

[11]https://www.ibiblio.org/pub/linux/docs/LuCaS/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisa-redes_neuronales-html/x152.html

[12] https://www.electroingenio.cl/microfono-max9814/

[13] arduino.cl/arduino-yun/

[14] https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

Anexo I

Código fuente de la fase de entrenamiento.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%% SELF-ORGANIZED MAP   %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
close all
%%
%% load data
%Carga de datos

datos=readtable('matlab.xlsx');

%% AdecuaciÛn para los vectores
fh=[];
d=[];
for iter=1:length(datos.s1)
    %FRANJA HORARIA
    aux_fh=[0 0 0 0];
    if datos.fh(iter)==1;aux_fh(1)=1;
    elseif datos.fh(iter)==10;aux_fh(2)=1;
    elseif datos.fh(iter)==100;aux_fh(3)=1;
    elseif datos.fh(iter)==1000;aux_fh(4)=1;
    end
    fh=[fh;aux_fh];
    %DIA
    aux_d=[0 0 0];
    if datos.d(iter)==1;aux_d(1)=1;
    elseif datos.d(iter)==10;aux_d(2)=1;
    elseif datos.d(iter)==100;aux_d(3)=1;
    end
    d=[d;aux_d];
end

datos = [datos.s1 datos.s2 datos.s3 fh d];

input = datos;
%% recogida de datos

[nn, natrib]=size(input); %% Collecting samples and atributes

names={'s1' 's2' 's3' 'fh' 'd'};

% Traspose convenient for nn training
input =input';

%% RepresentaciÛn de los datos recogidos
% figure(1)
```

```
%
% subplot(3,1,1)
% plot(1:nn, input(1,:))
% axis([0 840 0 4]);
% xlabel('Muestras')
% ylabel('Valores')
% title(names(1))
%
% subplot(3,1,2)
% plot(1:nn, input(2,:))
% axis([0 840 0 4]);
% xlabel('Muestras')
% ylabel('Valores')
% title(names(2))
%
% subplot(3,1,3)
% plot(1:nn, input(3,:))
% axis([0 840 0 4]);
% xlabel('Muestras')
% ylabel('Valores')
% title(names(3))

%% Crear el mapa
% building a map of size dimension1xdimension2

start=1;ending=nn;

dimension1 = 5; %Anchura del mapa - dimension X
dimension2 = 5;  %Altura del mapa - dimension Y

iterations = 2000;
radius_n=1;
top='hextop';
distance='dist';
net = selforgmap([dimension1 dimension2], iterations,radius_n, top,distance);

net.trainParam.lr = 0.6; % Factor de aprendizaje

% Entrenamiento de la red
[net,tr] = train(net,input(1:natrib, start:ending));

% Pesos
weights= net.IW{1};

%Redondear PESOS COLUMNAS SEMANAS
weights(:,8:10)=round(weights(:,8:10));

%% REDONDEAR PESOS COLUMNAS FRANJA HORARIA
 N=0;
for iter=1:1:25
        peso1 = weights (iter,4);
        peso2 = weights (iter,5);
        peso3 = weights (iter,6);
```

```
        peso4 = weights (iter,7);

        if (peso1 > peso2) && (peso1 > peso3) && (peso1 > peso4)
            weights (iter,4) = 1;
            weights (iter,5) = 0;
            weights (iter,6) = 0;
            weights (iter,7) = 0;
        elseif (peso2 > peso3) && (peso2 > peso1) && (peso2 > peso4)
            weights (iter,4) = 0;
            weights (iter,5) = 1;
            weights (iter,6) = 0;
            weights (iter,7) = 0;
        elseif (peso3 > peso1) && (peso3 > peso2) && (peso3 > peso4)
            weights (iter,4) = 0;
            weights (iter,5) = 0;
            weights (iter,6) = 1;
            weights (iter,7) = 0;
        else
            weights (iter,4) = 0;
            weights (iter,5) = 0;
            weights (iter,6) = 0;
            weights (iter,7) = 1;
    end
end

%% Plotting the patterns obtained
figure(1)
for i=1:dimension1*dimension2
    plot(1:natrib,weights(i,:))
    hold on
end
title('Pesos Atributos')
xlabel('Atributos')
ylabel('Valores Atributos')
legend('P1','P2','P3','P4','P5','P6','P7','P8','P9','P10','P11','P12','P13','P14','P15','P16','P17','P18','P1
9','P20','P21','P22','P23','P24','P25')
hold off

%% ORDEN DE LOS PATRONES
map_order=[21 22 23 24 25 16 17 18 19 20 11 12 13 14 15 6 7 8 9 10 1 2 3 4 5];

%%
figure(2)
for i=1:size(weights,1)
    subplot(dimension2,dimension1,map_order(i))
    b=bar(weights(i,:));
    c = b.FaceColor;
    %xlabel('Atributo')
    xticks([1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24])
    %ylabel('Valores Atributo')
    title(['PatrÛn ', num2str(i)])
end

%%
```

```
figure(3)
for i=1:size(weights,1)
    subplot(dimension2,dimension1,map_order(i))
    plot(weights(i,:))
    xlabel('Atributo')
    xticks([1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24])
    ylabel('Valores')
    title(['PatrÛn ', num2str(i)])
end

%%
% Checking the ranges of the patterns obtained
% The winner neuron is put to 1

training_set=input;

%ytrain=net(input(:,start:6:ending));
ytrain=net(training_set(1:natrib,start:ending));
hits_count= sum(ytrain,2); %% per pattern discovered

figure(4)
for i=1:size(weights,1)
    a=find(ytrain(i,:)); % indexes of the samples in each neuron
    subplot(dimension2,dimension1,map_order(i))
    histogram(training_set(natrib,a));
    xlabel('Dia')
    ylabel('Atributo')
    title(['PatrÛn', num2str(i)])
end
```

Anexo II

Código fuente de la fase de validación.

```
%% FASE DE VALIDACIóN
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% AdecuaciÛn para los vectores VALIDACIóN
datos_prueba=readtable('matlab_prueba.xlsx');
fh=[];
d=[];
for iter=1:length(datos_prueba.s1)
    %FRANJA HORARIA
    aux_fh=[0 0 0 0];
    if datos_prueba.fh(iter)==1;aux_fh(1)=1;
    elseif datos_prueba.fh(iter)==10;aux_fh(2)=1;
    elseif datos_prueba.fh(iter)==100;aux_fh(3)=1;
    elseif datos_prueba.fh(iter)==1000;aux_fh(4)=1;
    end
    fh=[fh;aux_fh];
    %DIA
    aux_d=[0 0 0];
    if datos_prueba.d(iter)==1;aux_d(1)=1;
    elseif datos_prueba.d(iter)==10;aux_d(2)=1;
    elseif datos_prueba.d(iter)==100;aux_d(3)=1;
    end
    d=[d;aux_d];
end

datos_prueba = [datos_prueba.s1 datos_prueba.s2 datos_prueba.s3 fh d];

input = datos_prueba;

%% Fase validaciÛn
prediccion= sim(net, input');
%% RepresentaciÛn de los datos recogidos
% [nn, natrib]=size(input); %% Collecting samples and atributeS
% names={'s1' 's2' 's3' 'fh' 'd'};
%
% input = input';
% figure(1)
%
% subplot(3,1,1)
% plot(1:nn, input(1,:))
% axis([0 840 0 4]);
% xlabel('Muestras')
% ylabel('Valores')
% title(names(1))
%
```

```
% subplot(3,1,2)
% plot(1:nn, input(2,:))
% axis([0 840 0 4]);
% xlabel('Muestras')
% ylabel('Valores')
% title(names(2))
%
% subplot(3,1,3)
% plot(1:nn, input(3,:))
% axis([0 840 0 4]);
% xlabel('Muestras')
% ylabel('Valores')
% title(names(3))
```