UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA
Ingeniería de Telecomunicación

DESARROLLO DE UNA PLATAFORMA PARA DISCRIMINACIÓN DE ODORANTES MEDIANTE TÉCNICAS DE MODULACIÓN DINÁMICA.

Sergio de la Cruz Gutiérrez
TUTOR: Francisco de Borja Rodríguez Ortiz

JULIO 2016

DESARROLLO DE UNA PLATAFORMA PARA DISCRIMINACIÓN DE ODORANTES MEDIANTE TÉCNICAS DE MODULACIÓN DINÁMICA.

AUTOR: Sergio de la Cruz Gutiérrez
TUTOR: Francisco de Borja Rodríguez Ortiz

GNB
Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2016

Resumen

Este proyecto plantea la realización de una plataforma de experimentación para la adquisición y discriminación de odorantes. La investigación en el ámbito de las narices electrónicas y sus posibles usos para discriminación y clasificación de odorantes son actualmente un campo activo de investigación. La plataforma debe ser versátil, fácil de programar y de tamaño reducido. En cuanto a su versatilidad debe de ser capaz de aplicar distintas técnicas de modulaciones para la detección de odorantes, bien sean en amplitud o en frecuencia.

Para la consecución de este objetivo utilizamos sensores de tipo MOS y estudiamos cómo influirá la elección de un sistema de control, que sea intuitivo y fácil de manejar. En este aspecto se estudiarán las diversas posibilidades que ofrece el mercado, desde microprocesadores hasta sistemas embebidos. Se creará un sistema de flujo desde el odorante al sensor y se diseñará la estructura que albergue los distintos componentes de la plataforma.

La plataforma contendrá tres sensores de odorante de tipo MOS, cada uno con su propio circuito de adaptación y sus correspondientes códigos ejecutables de experimentación. Se realizará un circuito, denominado puro, en el cual se realizará la simple adquisición de muestras de odorante sin modulación alguna. Un segundo circuito, denominado de modulación de amplitud, el cual se usará en experimentos en los que se realice modulación por amplitud. Y un tercer circuito para modulaciones por frecuencia.

Con la finalización del proyecto se ha logrado desarrollar una plataforma de investigación e experimentación, en el contexto de narices electrónicas, que permite realizar experimentaciones con y sin técnicas de modulación tanto en frecuencia como en amplitud. El desarrollo de esta plataforma deja la puerta abierta para introducir nuevos tipos o técnicas de modulación, pues se ha realizado de forma que sea fácil de seguir desarrollándola. Se han desarrollado protocolos de experimentación versátiles que son muy fáciles de programar en Python.

Como resultado final en este proyecto se presentan unos ejemplos del funcionamiento y utilización de la plataforma, que integra diferentes metodologías de modulación.

Palabras Clave

Nariz artificial, BeagleBone, plataforma de experimentación, modulación de temperatura de calentamiento, sensores MOS, adquisición de odorante, Python, hardware libre.

Abstract

This project proposes the production of an experimental platform for the acquisition and discrimination of odorous. The investigation in the field of electronic noses and its possible uses for discrimination and classification of odorous are currently an active investigation field. The platform has to be versatile, easy to program and of a reduced size. Regarding its versatility, it has to be able to apply different modulation technics for the detection of odorous, either in amplitude or frequency.

For the fulfilment of this objective MOS type sensors have been used and it has been studied how the decision of an intuitive and easy to use control system effects on performance. On this matter, the different possibilities which the market offers will be studied, going from microprocessors up to embedded systems. A flow system will be created from the odorous to the sensor, and the structure which will harbor the different components in the platform will be also designed.

The platform will consist of three odorous sensors MOS type, each with its own adaptation circuits and in its corresponding experimental executable codes. A circuit, named pure, will be created to perform the acquisition of odorous samples without any modulation. A second circuit, named amplitude modulation, will be used in experiments in which amplitude modulation will be carried out. Finally, and on the same basis, a third circuit will be created for frequency modulations.

When finished the Project, it has developed a platform for research and experimentation, in the context of electronic nose, which will allow to perform experiments with and without modulation technics. The development of this platform leaves the door open to the introduction of new modulation techniques, as this platform has been done so it easy to continue developing. A versatile experimental protocols has been developed to be easy to program in Python.

As a final result, in this project, some examples of operation and use of the platform, Which Integrates different modulation methods are presented.

Keywords

Electronic nose, BeagleBone Black, experimental platform, modulation heating temperature, MOS sensors, acquisition of odorant, Python, free hardware.

Agradecimientos

En primer lugar, quiero agradecer por el apoyo y confianza depositada en mi al confiarme este proyecto a mi tutor, Francisco de Borja. Agradecer su confianza, dedicación y compromiso, pues sin él no habría sido posible este proyecto. Igualmente, dar las gracias a Pablo Varona, quien junto a Francisco me abrió las puertas del GNB y me han llevado a desarrollar mi faceta investigadora.

Agradecer, cómo no, a mis padres, Pilar y Jerónimo, quienes desde que era pequeño se han desvivido por mí y siempre han luchado para nunca me falte de nada. Por su esfuerzo y perseverancia en mi educación, sin ellos es probable que no hubiese llegado hasta aquí, así que, ¡Gracias! A mí, hermana Paula, por esos veintiún años de risas y discusiones, ¡estudia, que la próxima eres tú!

Como olvidarme de los compañeros de fatigas durante todos estos años, Hugo, Carlos, More, Rubén, Andrés y todos los que faltan, son muchos y de todos he aprendido algo. Agradecerles su tiempo, su compañía y amistad durante estos años, gracias a ellos esta etapa se ha hecho mucho más llevadera.

Quiero acordarme también de mi etapa erasmus, agradecer a mis padres esa oportunidad, pero sobretodo, dar las gracias a todos con los que compartí experiencias ese año. De todos y cada uno de ellos aprendí algo, me ayudaron a madurar.

Índice de Contenidos

Índice de figuras

Índice de Tablas

1. Introducción

1.1. Motivación

El olor es la sensación resultante de la recepción de un estímulo por el sistema sensorial olfativo. El término abarca tanto la impresión que se produce en el olfato, como lo que es capaz de producirlo. El olor es generado por una mezcla compleja de gases, vapores y polvo, donde el olor percibido está influido directamente por la composición de la mezcla. En adelante, a esta mezcla la denominaremos odorante.

En un sistema olfativo un odorante es captado por los receptores, localizados en el epitelio de la parte superior de la cavidad nasal. Los receptores transforman el estímulo en una señal que es enviada al cerebro, para que este la identifique. Los sistemas olfativos artificiales nacen con el objetivo de reproducir la capacidad de lectura e interpretación de estas señales.

Las narices artificiales (EN's, del inglés Electronic Nose) son máquinas diseñadas con el fin de detectar y discriminar con precisión distintos odorantes. Una definición generalmente aceptada de un sistema olfativo fue dada por Gardner y Barlett en 1994: "*instrumento que comprende una agrupación de sensores químicos con sensibilidades parcialmente solapadas junto a un sistema de reconocimiento de patrones, capaz de analizar y reconocer aromas simples o complejos*".

En Grupo de Neurocomputación Biológica (GNB) ha dedicado un área de investigación a los sistemas olfativos artificiales. Distinguiéndose dos claras ramas de estudio, localización de la fuente de odorante y discriminación de odorantes:

- El trabajo de final de máster realizado por David Yañez "*Estrategias bioinspiradas para la adquisición de olores en narices artificiales*". Integra un equipo de nariz artificial capaz de soportar sensores resistivos multicanal (TGS) [1]. Que ha dado lugar a posteriores publicaciones relacionadas con él [2][3][4].

- El trabajo de final de máster de Tomás Vázquez supuso el inicio de los estudios de las aplicaciones de los sensores de odorante integrados en robótica [5].

- El trabajo final de máster de Carlos García-Saura plantea un enfoque de localización de odorante mediante la cooperación de varios robots, que utilizan lógicas de localización para una única nariz artificial [6].

- Posteriormente Alejandro Pequeño Zurro, en su trabajo final de carrera [7], presenta la localización de la fuente de odorante con estrategias multisensor de narices electrónicas integradas en una única plataforma.

Si bien, trabajos anteriores del GNB se han enfocado tanto en la localización de la fuente de odorante como en la discriminación del odorante. En esta memoria de proyecto se aborda la discriminación de odorante con el uso de sensores de bajo coste. Se tratará de diseñar una plataforma de discriminación tomando como modelo la elaborada por GNB [1]. Esta utilizaba un PIC como sistema de control para la adquisición y discriminación del odorante, lo que la hacía poco versátil y difícil de programar para alguien que no sea experto en ello. Aunque se tome como idea la plataforma diseñada anteriormente por el GNB, este proyecto partirá de cero, pues uno de los motivos principales del mismo es que la plataforma sea fácil de utilizar para alguien que no sea experto en la materia y por ello la elección de los componentes, en especial el sistema de control, se realizará de nuevo. Actualmente hay microprocesadores muy potentes o sistemas embebidos que permiten programar en lenguajes de alto nivel y una comunicación entre entradas y salidas más intuitiva que con la utilización de un PIC.

La plataforma creada por el GNB [1] distinguía odorante utilizando la variación de voltaje que genera un cierto tipo de gas en un sensor de odorante de bajo coste tipo TGS[1][2][10]. De forma que, estudiando los resultados obtenidos al realizar diversos experimentos con distintos tipos de odorantes, poder llegar a un patrón de detección. Posteriormente, se publicaron artículos que plantean la posibilidad de utilizar técnicas de modulación para una mejor detección y discriminación, en concreto [3], donde se realiza modulación en amplitud por seguimiento de temperatura.

La publicación realizada por el grupo de GNB [3] presenta una modulación de temperatura para la discriminación del odorante, modulación enfocada a la amplitud de la señal. Sin embargo, esta plataforma adolece de la modulación en frecuencia. Una de las principales motivaciones de este proyecto es incluir las técnicas utilizadas en las publicaciones realizadas por Eugenio Martinelli [8][9] que introducen el uso de multivibradores como método de discriminación, utilizando como patrones de clasificación las variaciones en frecuencia que sufre la señal captada por el sensor. Más adelante se detallarán con exactitud estos mecanismos de detección.

A lo largo del presente proyecto se integrarán las diversas técnicas de discriminación, comentadas anteriormente, en una misma plataforma de experimentación que permita explotar al máximo las características dinámicas de la onda generada por un sensor de odorante de bajo coste. Sin descartar, la posible introducción o prueba de nuevas técnicas de modulación en la misma además de las mencionadas. En la industria ya existen sensores capaces de detectar odorantes, pero el problema de estos es su alto coste. Con el presente trabajo se pretende lograr una plataforma de detección fiable, robusta y barata; que sirva para facilitar la experimentación en discriminación de odorante. Se creará una plataforma más flexible y versátil, que permita integrar varios tipos de modulación y que sirva para futuras experimentaciones con odorantes y nuevas técnicas de modulación.

1.2. Objetivos

En la presente sección se exponen los objetivos globales y las tareas que marcaron el desarrollo del proyecto.

El objetivo general es la elaboración de una plataforma de experimentación versátil, que pueda aplicar diferentes modulaciones para la detección de odorantes. Que sea fácil de programar y de tamaño reducido. La consecución del objetivo principal lleva ligada una serie de tareas u objetivos secundarios que se exponen a continuación:

1. Estudio, análisis y comprensión de los trabajos previos realizados en el departamento del GNB [2][3][4][5][6][7].

    a. Estudio del mecanismo de detección utilizados en [1], al igual que el funcionamiento de la plataforma.

    b. Lectura y estudio de los diferentes protocolos de comunicación entre sensores y controlador.

2. Lectura y estudio de la literatura sobre narices artificiales.

3. Búsqueda y estudio de documentación, trabajos y publicaciones realizadas sobre técnicas de modulación en narices artificiales.

    a. Estudio y compresión de las técnicas de adquisición y discriminación de odorante realizadas en [1].

    b. Estudio y compresión de las técnicas de modulación en frecuencia realizadas en [8][9].

    c. Estudio y comprensión de las técnicas de modulación por variación de temperatura realizadas en [3].

    d. Búsqueda de información general y comprensión sobre técnicas de modulación en "*Handbook of Machine Olfaction"* [11].

4. Estudio general de sustancias a detectar.

5. Elaboración de la plataforma de detección:

    a. Estudio y entendimiento de la plataforma de David Yañez [2].

    b. Estudio de los diferentes tipos de sensores disponibles en el mercado. Comparación entre ellos y elección del sensor más adecuado para la plataforma.

    c. Estudio y comparación entre las plataformas de hardware libre disponibles en el mercado (Arduino, Rapsberry y Beaglebone). Elección de la más adecuada para su integración en la plataforma.

    d. Estudio y comparación de componentes para la realización de un circuito de adquisición de odorante.

e. Estudio y diseño de circuitos electrónicos para la adaptación de señales eléctricas.

f. Búsqueda y elección del resto de componentes a integrar en la plataforma. Condensadores, resistencias, electroválvulas, motor de succión, transistores, transformadores, etc.

g. Estudio de la disposición de los componentes en la plataforma, con el fin de crear una plataforma de tamaño reducido, de fácil acceso a sus partes y con las menores distancias posibles entre componentes, para así reducir errores en las medidas.

6. Estudio y familiarización con el entorno de programación de la plataforma elegida (Python, C++). Desarrollo del software del sistema.

7. Pruebas con la plataforma de detección.

8. Estudio de los resultados obtenidos.

9. Evaluación del funcionamiento del sistema. Conclusiones.

1.3. Organización de la memoria y metodología utilizada

La presente memoria consta de los siguientes capítulos:

1. **Introducción.** Introducción al proyecto, motivación, objetivos y organización de la memoria. Es el capítulo en el que nos entramos actualmente.

2. **Estado del arte.** En este capítulo se expone el marco teórico en el que se encuentra el proyecto, se expone de forma más precisa el problema a tratar y las soluciones al mismo. Estudio de las narices electrónicas, estado actual de los sistemas de discriminación de odorantes. Introducción a las técnicas de modulación dinámica.

3. **Estudio y elección de componentes de la plataforma.** En esta sección se detallará la elección realizada de componentes principales de la plataforma. Desde la elección del sistema de control hasta los sensores de odorante utilizados, pasando por otros dispositivos necesarios, por ejemplo, para monitorizar la temperatura o humedad. No sólo se referenciarán los componentes utilizados si no que se comparan con otros disponibles en el mercado y se expondrán los pros y contras que llevan a su elección.

   Como la plataforma será capaz de integrar tanto modulaciones en frecuencia como en amplitud, será necesario elaborar dos circuitos de adaptación distintos para cada modulación.

4. **Metodología y tecnologías utilizadas.** En esta sección se detallan los pasos seguidos para la creación de la plataforma de experimentación, desde el diseño de la estructura de montaje hasta la configuración de la placa BBB. Pasando por el arranque de la BBB por primera vez, una breve introducción al lenguaje de programación que se va a utilizar o las librerías necesarias.

5. **Desarrollo de la plataforma.** Se explica el desarrollo de la plataforma tanto a nivel de hardware como de software. Se exponen los circuitos de adaptación diseñados para la adquisición en función de si la modulación es por amplitud o por frecuencia. Además del resto de circuitos de potencia necesarios para el funcionamiento de las electroválvulas y el motor de succión. En la parte software se explican los códigos elaborados y se detallan las técnicas utilizadas para el acceso a los pines de entrada y salida, así como los nombres y ruta de los ficheros de salida de resultados y el funcionamiento de las funciones principales.

6. **Resultados y validación.** En esta sección se comprobará que los códigos de experimentación elaborados, así como los circuitos creados cumplen con los objetivos del proyecto. Se validará que todas las señales de entrada en la placa estén dentro del margen adecuado de trabajo y que los resultados que se obtienen de los distintos experimentos sean válidos para su estudio y discriminación de odorante en futuros proyectos del GNB.

7. **Conclusión y trabajo futuro.** Último capítulo de la memoria de este proyecto, conclusiones del trabajo realizado, consecución de objetivos y perspectivas de trabajo futuro para el desarrollo de este proyecto y otros derivados del mismo.

2. Estado del arte

2.1. Introducción

Antes de introducirnos en el diseño que se va a llevar a cabo en el proyecto conviene introducir al lector en ciertos aspectos que le ayudarán a entender el porqué de este trabajo. Poniendo al lector en una posición desde la cual sea capaz de entender los temas que se van a desarrollar en los próximos capítulos.

El olor es una sensación que ocurre cuando los receptores localizados en el epitelio olfativo de la parte superior de la cavidad nasal son estimulados con ciertas sustancias químicas que se encuentran en el aire. Las narices electrónicas (EN's) son máquinas elaboradas para efectuar la detección y discriminación con precisión de olores.

Los orígenes de la nariz electrónica se remontan a los años 60, cuando la compañía *Bacharach Inc*, construyo un primer dispositivo. En la década de los 80, surgen dos nuevos grupos de investigación, en la Universidad de Warwick en Gran Bretaña y en el Argonne Nacional Laboratory (ANL) en Estados Unidos, debiéndose las primeras publicaciones a Krishna Persaud y George Dodd (1982). Orientaron sus estudios a entender los procesos del olfato biológico, utilizando un conjunto de sensores semiconductores de óxidos metálicos. Posteriormente en Japón se comenzó a investigar la frescura de los pescados utilizando matrices de sensores semiconductores de óxido metálico.

Todos estos avances han dado lugar a la fabricación comercial de narices electrónicas, se pueden ver algunas de ellas en la sección 7 de libro "*Handbook of Machine Olfaction"* [11]. La mayoría de las diseñadas hasta la fecha son para uso comercial, algunas otras son utilizadas por los gobiernos para seguridad, y otras son usadas en equipos hospitalarios. En la actualidad, las EN's no son solamente usadas para clasificaciones de aroma, sino también para la detección de olores. De ahí que su importancia y utilización se haya extendido a aplicaciones en la agroindustria, medio ambiente, la seguridad, y la medicina.

2.2. Nariz electrónica frente Sistema olfativo humano

El sentido del olfato inicia su funcionamiento cuando se ve estimulado por moléculas que se encuentran en el aire o por sustancias volátiles que contienen los alimentos que nos llevamos a la boca.

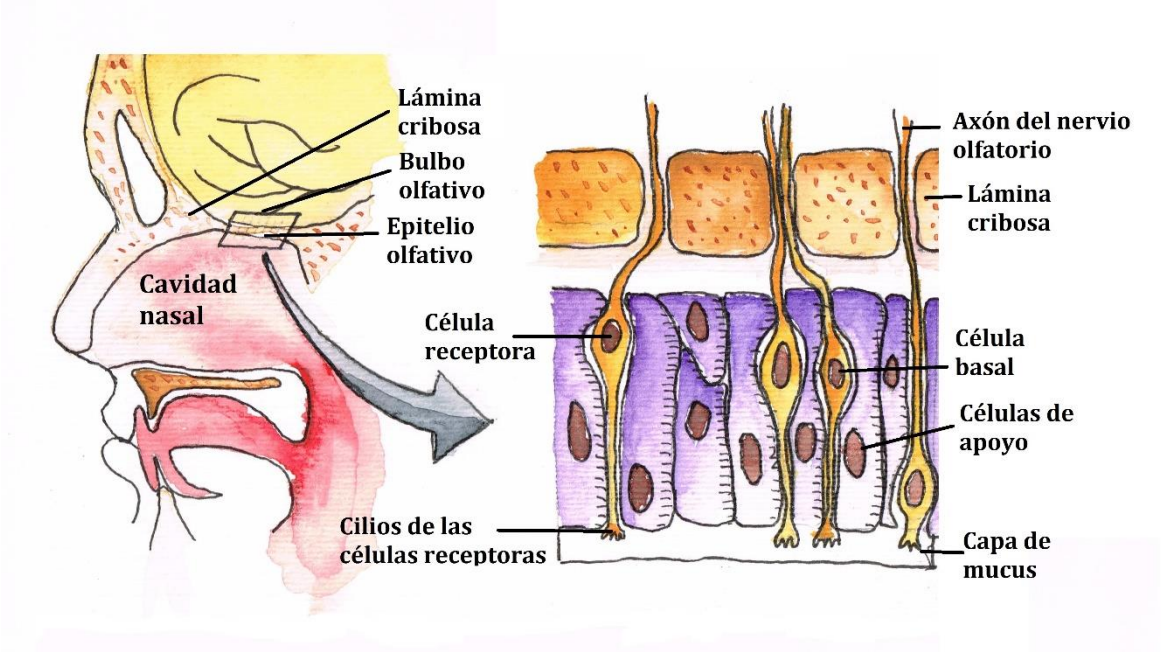


Figura 2. 1: *Representación del sentido olfativo. Epitelio olfativo.*

El sentido del olfato comienza en la nariz, este sentido en sistemas biológicos se articula mediante receptores sensoriales que se encuentra en el epitelio olfativo, Figura 2. 1. Análogamente los sensores artificiales creados para dicho propósito poseen unas superficies sobre las que el componente químico excita al sistema, como puede observarse en la Figura 2. 2. Además, se ven implicadas otras partes de la cabeza y el cerebro como son las fosas nasales, las neuronas receptoras del olfato, el bulbo olfativo y el cerebro, que es donde se realiza el proceso de identificación de un olor.

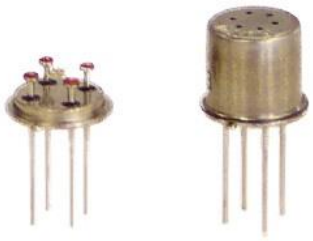


Figura 2. 2: *Sensor modelo TGS de la casa Fígaro (Imagen adaptada del datasheet del componente)* [10].

Cada olor que hay en la naturaleza es una mezcla de sustancias químicas y tiene sus propias características, que lo diferencian de los otros olores. A pesar de que los seres humanos tienen una sensibilidad limitada (en comparación con algunos animales) son capaces de percibir, identificar y clasificar olores. El recorrido de este proceso en el sistema

del olfato humano, comparado con el de la nariz electrónica se puede observar en la Figura 2. 3.

Las moléculas de olor son expuestas a la nariz electrónica, los patrones químicos presentes en la muestra de aroma son detectados por los sensores, los cuales transforman esta entrada química en una señal eléctrica produciendo para cada aroma un único patrón de respuesta, designado como huella digital olfativa. Finalmente, a esta respuesta se le aplican técnicas de reconocimiento de patrones para discriminar, clasificar y predecir el tipo de aroma que se está analizando.

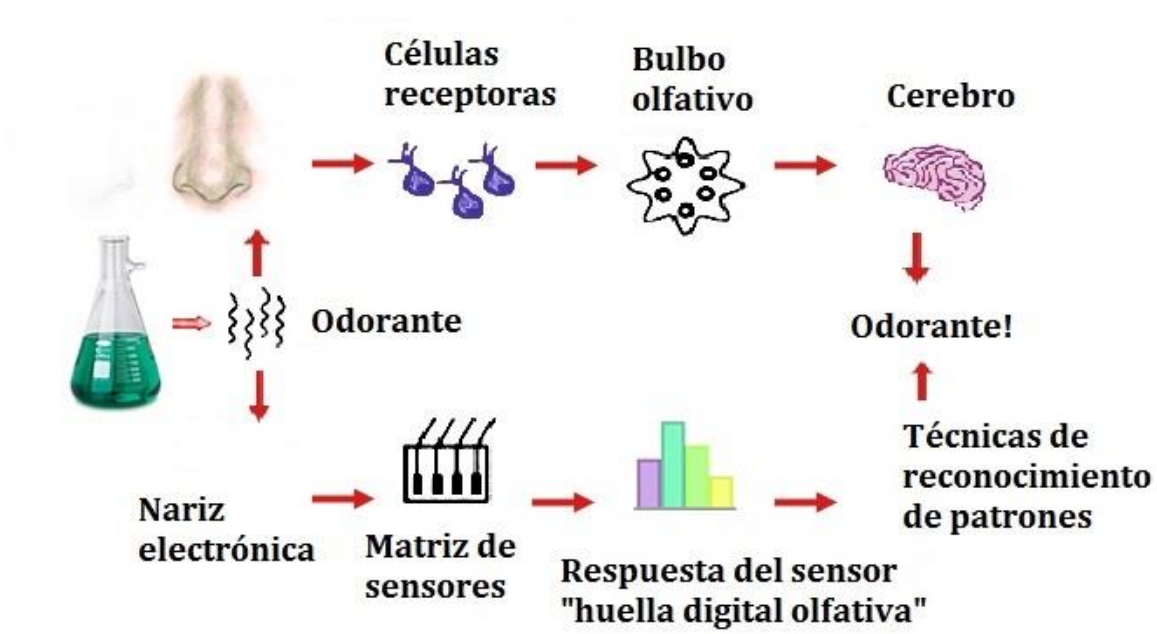


Figura 2. 3: *Ejemplo de nariz electrónica frente a sistema olfativo humano.*

En las narices electrónicas se distinguen generalmente tres módulos: de detección, electrónico y de procesamiento. En el módulo de detección se incluyen los sensores encargados de traducir de manera directa o indirecta la presencia de un olor en señales eléctricas. Cuando entran en contacto con componentes volátiles presentes en el gas, los sensores reaccionan, experimentando un cambio físico en sus propiedades. La respuesta del sensor se recibe en el módulo electrónico, que transforma la señal en un valor digital. Los valores se analizan a continuación en el módulo de procesamiento, que se encarga de realizar el reconocimiento y/o la clasificación de las señales registradas.

El método de funcionamiento de una nariz artificial comprende una etapa de aprendizaje o entrenamiento, en la que se somete a la nariz artificial al análisis de una serie de muestras reconocidas para construir modelos de referencia. En una etapa posterior de adquisición del olor, la nariz artificial es capaz de reconocer nuevas muestras a partir de los modelos de referencia.

Existen muchos tipos de sensores olfativos basados en diferentes principios físicos:

- **Quimiorresistores.** Sensores que varían su conductividad eléctrica ante la presencia de ciertas sustancias químicas. Son bajamente selectivos por lo que la identificación de un único químico es difícil, pero son económicos y de tamaño reducido. Es este tipo el que utilizaremos para el proyecto [10].

- **Quimiotransistores.** Transistores con un recubrimiento que los hacen sensibles a sustancias químicas. Pueden usarse en entornos líquidos y gaseosos.

- **Quimiocapacitore**s**.** El sensor es un condensador cuyo dieléctrico es un polímero que varía su capacitancia cuando las moléculas gaseosas a analizar son absorbidas.

- **Quimiosensores amperimétricos.** Miden la corriente eléctrica que pasa por una celda electroquímica. Son muy sensibles, pero de coste elevado.

		Principio	Medición	Tipo de Sensor
Directos	**Físicos**	Conductómetro	Conductancia	Químico Resistivo (CP)
		Capacitivo	Capacitancia	Químico Capacitivo -Polímero
	Químicos	Amperimetría	Corriente	Gas Tóxico -Electrocatalítico
		Conductométrico	Conductancia	Químico Resistivo -MOS
		Potenciométrico	Voltaje/emf	Químico Diodo - Diodo Schottky
			I-V/C-V	Químico Transistor - MOSFET
Complejos	**Físicos**	Gravimétrico	Piezoelectricidad	Químico sensible a masa - QCM
				Químico sensible a masa - SAW
		Óptico	Intensidad/Espectro	Químico de fibra óptica - Fluorescencia, quimioluminiscencia
	Químicos	Calorimétrico	Temperatura	Químico Térmico - Termistor, Termopar
		Amperimetría, témico, óptico	Corriente, temperatura, intensidad de luz	Biosensores.

Tabla 1: *Clasificación de sensores de Olor.*

2.2.1. Sensores Quimiorresistivos

Son los sensores más utilizados debido a su pequeño tamaño, su fácil integración en un circuito eléctrico y su bajo coste. Dichos sensores son los comúnmente utilizados para narices electrónicas, siendo los basados en Óxidos Metálicos Semiconductores (MOS) los más usados dentro de esta categoría.

Los sensores basados en MOS, se usan desde 1960 como sensores de gases en las alarmas de incendios [11]. Consisten en un sustrato cerámico cubierto de una película del óxido semiconductor. Este óxido se puede depositar mediante diversas técnicas como la

evaporación, la técnica CVD (*Chemical Vapor Deposition*) [11], las técnicas de spray, etc. El óxido metálico [12][14] puede ser de tipo n (óxido de zinc, de estaño, dióxido de titanio u óxido de hierro) y responde a compuestos oxidantes, o de tipo p (óxido de níquel o de cobalto) y a gases reductores.

Para conseguir sensores con diferente selectividad hacia distintos compuestos químicos, la película de óxido metálico se dopa con metales nobles catalíticos como platino o paladio. También es posible variar su selectividad modificando el tamaño del grano del semiconductor policristalino.

La Figura 2. 4. muestra la estructura básica de un sensor MOS de la serie TGS2600 [10]. Como se puede apreciar en la figura el sensor está compuesto por una base sobre la cual se encuentra el elemento de detección que contiene el material sensible, con una resistencia calefactora que regula automáticamente su temperatura óptima de funcionamiento. Todo el dispositivo se encuentra protegido por una cápsula que permite el paso de los gases al interior del sensor y previene su contaminación.

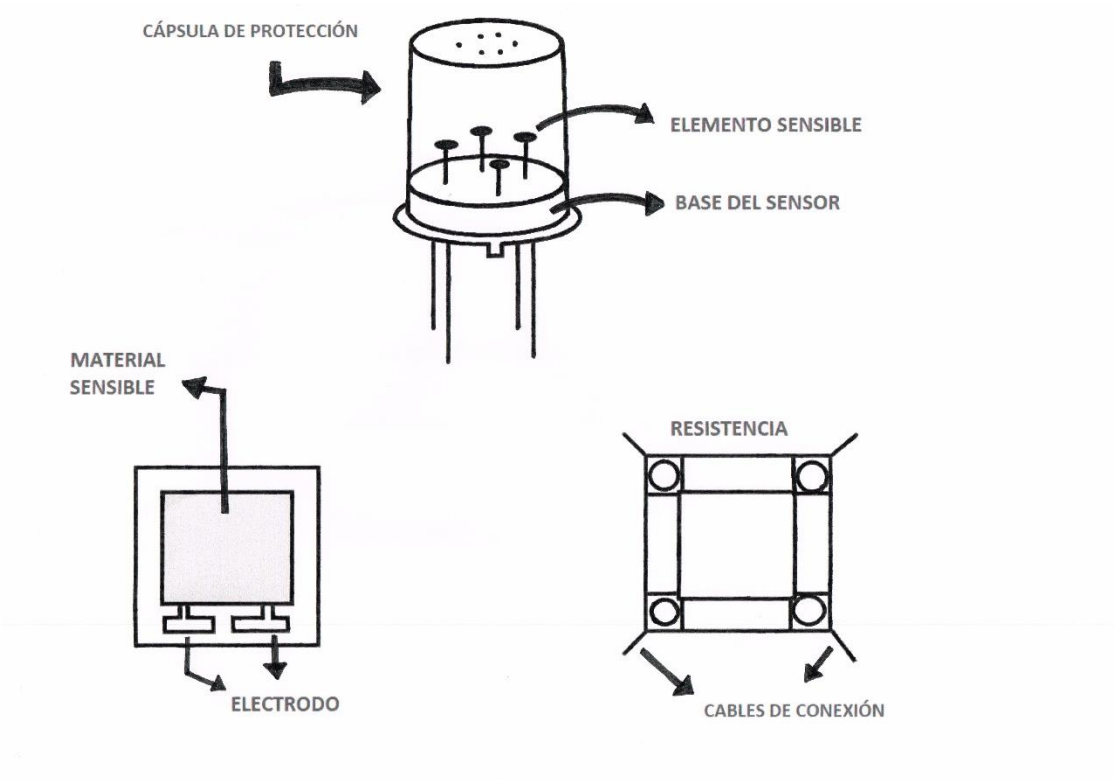


Figura 2. 4: *Estructura del sensor.*

Los detalles en profundidad sobre los sensores químicos y su principio de funcionamiento pueden ser consultados en las referencias que se citan a continuación [12][13][14][16]. A manera de resumen, el principio de detección de gases de estos sensores se basa en una reacción mediada por el óxido de la superficie y el oxígeno ionizado absorbido con las moléculas oxidantes o reductoras de la muestra que está siendo detectada. La respuesta de los sensores es la medida del cambio en la resistencia entre dos electrodos como un resultado de las reacciones químicas en la superficie del semiconductor del óxido metálico.

Existen dos tipos de sensores de óxidos metálicos:

- Capa gruesa: Se fabrican colocando el óxido metálico en forma de pasta entre los electrodos. Su principal ventaja es su facilidad de fabricación, por tanto, su bajo coste. Pero tiene varias desventajas, como que son muy poco selectivos y que su respuesta depende en gran medida de la temperatura y humedad del ambiente. Además, tienen un periodo transitorio muy largo, tardan mucho en estabilizar su respuesta.

- Capa fina: Tiene un proceso de fabricación distinto, mediante deposición de vapores. Esto permite obtener películas muy delgadas de óxido metálico entre los electrodos. La principal ventaja es que la respuesta de estos sensores es mucho más específica y consumen menos electricidad, pero resultan mucho más caros y difíciles de producir.

A continuación, se expone una breve explicación del funcionamiento de un sensor quimiorresistor:

a) En condiciones de aire libre de contaminante (con un 21% O_2) el oxígeno se acumula en la superficie del sensor. La presencia de oxígeno genera que los electrones libres de la superficie del sensor se vean atraídos creando así un potencial de barrera (*eVs* para el aire). Lo que hace que se genere una resistividad muy elevada, dado que el flujo de electrones se detiene. Este hecho se puede observar en la Figura 2. 5.

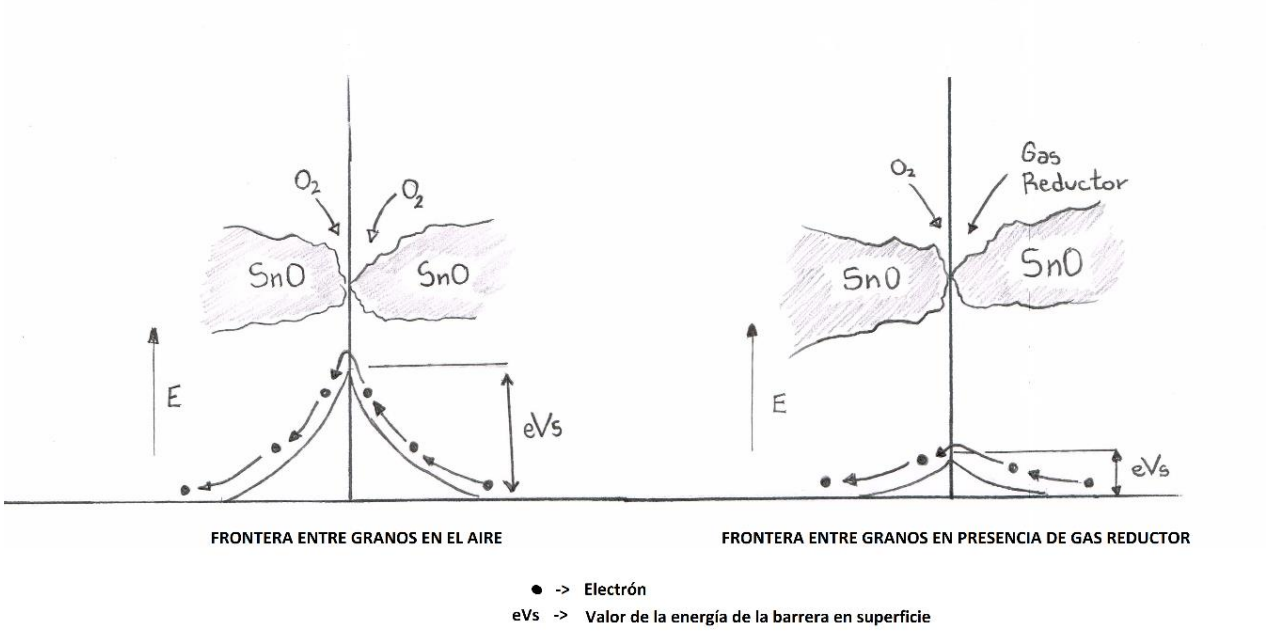


Figura 2. 5: *Interacción del sensor con la atmósfera. Figura izquierda en aire libre de contaminante. Figura derecha en presencia de gas reductor. Imagen reproducida de [34].*

b) La presencia de un gas contaminante hace que las partículas de oxígeno de la superficie del sensor se eliminen. Como consecuencia, la densidad del oxígeno en la superficie del sensor se reduce y con ello el potencial de barrera también. Una

disminución del potencial de barrera implica un aumento en el flujo de electrones y con ello un descenso en la resistividad del sensor. Por lo tanto, una medida directa de la resistencia del sensor nos da información tanto del gas como de la concentración de partículas del mismo. Este hecho se puede observar en la Figura 2. 5.

El siguiente esquema de imágenes presenta el funcionamiento de un sensor quimiorresistivo en dos simples pasos:

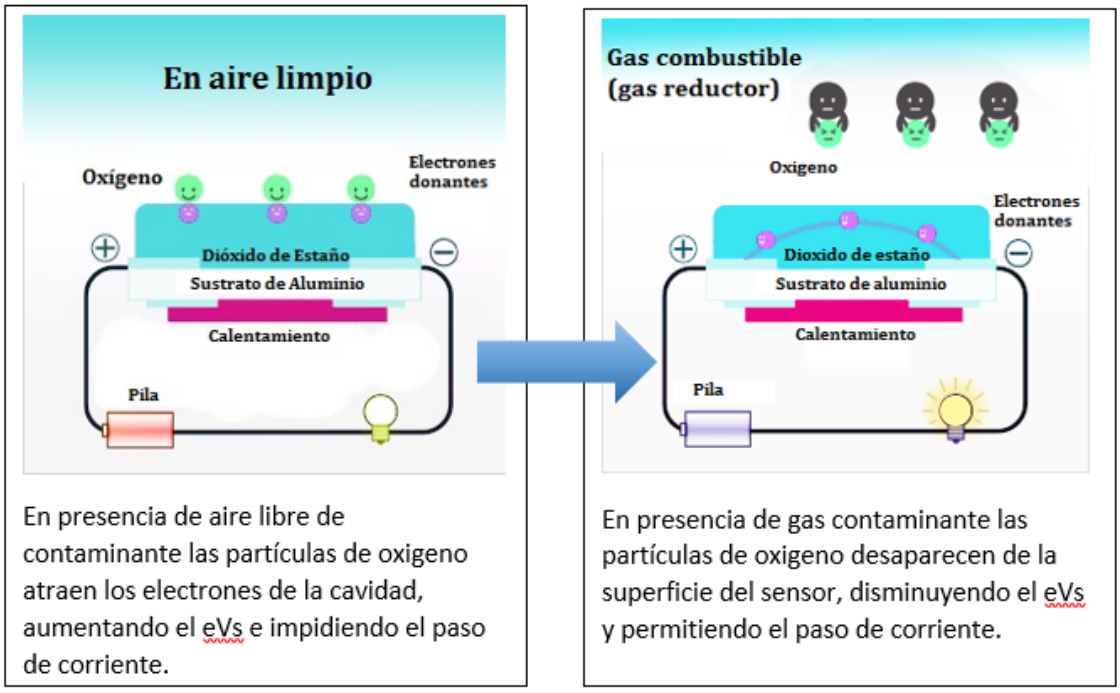


Figura 2. 6: *Funcionamiento del sensor quimiorresistivo. Imagen reproducida de [10]*

La señal de salida se obtiene por un cambio en la conductividad del óxido causado por la reacción entre el oxígeno absorbido y las moléculas entrantes en la capa sensible. La relación entre la resistencia del sensor y la concentración del gas reductor puede ser descrita para ciertos niveles de concentraciones en gases, a través de la siguiente ecuación [11]:

$$R_s = A[C]^{-\propto}$$

Dónde: R_s, resistencia eléctrica del sensor
A, constante
$[C]$, concentración del gas
α, constante experimental obtenida en función del gas evaluado

2.3. Técnicas de modulación

En Telecomunicaciones el término modulación engloba el conjunto de técnicas para transportar información sobre una onda portadora, típicamente una onda senoidal, en nuestro caso será la señal medida sensor. Básicamente, la modulación consiste en hacer que un parámetro de la onda portadora cambie de valor de acuerdo con las variaciones de la señal moduladora, que es la información que se quiere transmitir.

Existen tres aspectos básicos de la onda portadora que pueden modularse:

- Amplitud
- Frecuencia
- Fase o ángulo

Y las técnicas de modulación correspondientes, (aunque existen más):

- Amplitud modulada (AM)
- Frecuencia modulada (FM)
- Modulación de fase (PM)

Modulación en amplitud (AM): es un tipo de modulación lineal que consiste en varia la amplitud de la onda portadora de forma que esta cambie de acuerdo con las variaciones de nivel de la señal moduladora, que es la información que se va transmitir.

Dicha modulación consiste en modificar la amplitud de una señal de alta frecuencia, denominada portadora, en función de una señal de baja frecuencia, denominada moduladora, la cual es la señal que contiene la información que se desea transmitir.

Modulación en frecuencia (FM): es un tipo de modulación angular que transmite información a través de una portadora variando su frecuencia.

En el presente proyecto se pretende hacer uso de la modulación para sacar un mayor provecho a la información proporcionada por el sensor de odorante. En investigaciones anteriores del GNB ya se han realizado técnicas de modulación con este fin. En concreto, las investigaciones realizadas por David Yañez, en su trabajo fin de master [1]. Donde se presenta una EN's de detección que discrimina odorante mediante modulación de la amplitud de la señal recibida por el sensor, en función de la temperatura de calentamiento del mismo.

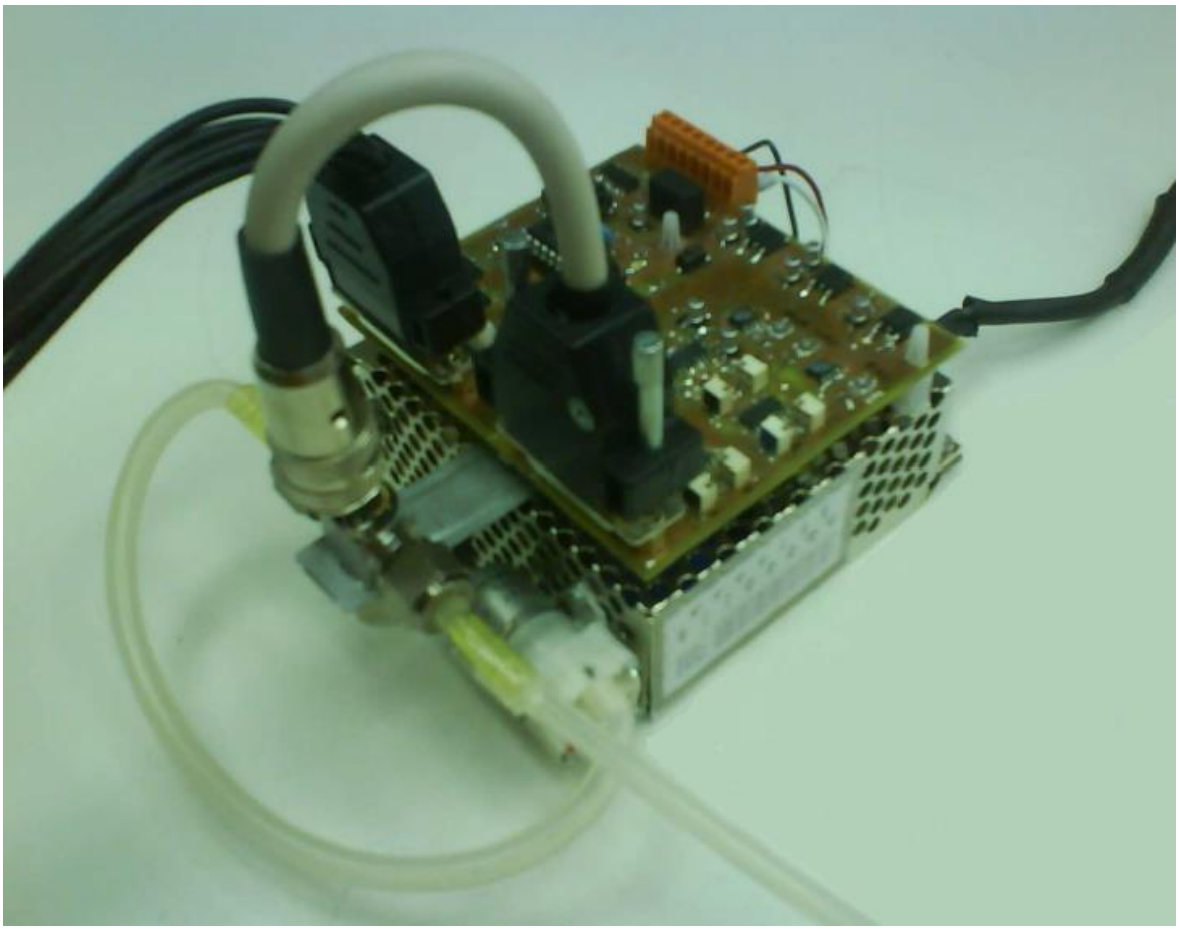


Figura 2. 7: *Sistema de nariz electrónica por modulación de amplitud. Imagen modificada de [1].*

La nariz electrónica de la Figura 2. 7 sirvió de inicio para posteriores investigaciones del GNB, siendo también de alta importancia para la realización del presente proyecto, la posterior publicación realizada también por David Yañez, "*An active, inverse temperatura modulation strategy for single sensor odorant classification*" [2]. Donde a partir de la nariz, antes citada, se creó una plataforma de experimentación para la detección y clasificación de odorantes.

La plataforma permite la entrada de un odorante cada vez en el sensor. Por medio de un sistema de electroválvulas y con la ayuda de un motor de succión, se conmuta el circuito de circulación del odorante para que al sensor llegue el odorante deseado. Por medio de una placa microcontroladora se lee la señal generada por el sensor, y haciendo uso de la técnica de modulación de temperatura (que se detallará más adelante) se discrimina el odorante.

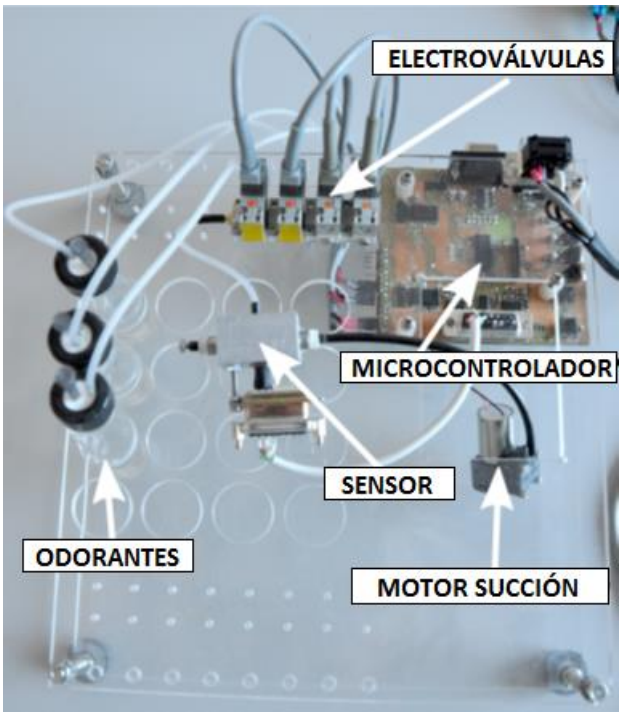


Figura 2. 8: *Plataforma experimental del GNB, imagen modificada de [3].*

Ahora, como se indicó en la sección [1.2], el objetivo principal de este proyecto es la creación de una nueva plataforma de experimentación más versátil y de fácil programación. Y que integre varias técnicas de modulaciones que sean desarrollables y que permita incluir otros métodos de modulación de manera sencilla. Aunque se va a partir desde cero para la realización de la plataforma, hay ciertos aspectos que se pretenden mejorar con respecto a la plataforma de David Yañez:

- Reducir su tamaño y así la distancia entre odorante, electroválvulas y sensor.
- Actualizar los componentes. Principalmente la placa microcontroladora, actualmente existen en el mercado placas muy potentes a bajo coste que permiten utilizar lenguajes de programación más intuitivos como Python o C++.
- Introducir nuevas técnicas de modulación. La realización de este proyecto nace con la idea de actualizar la plataforma del GNB para introducir la técnica de modulación en frecuencia propuesta por Eugenio Martinelli en "Self-adapted temperature modulation in metal-oxide semiconductor gas sensors" [8].
- Y lo más importante, que la plataforma a realizar sirva para futuros trabajos del GNB en la experimentación con narices electrónicas.

Se pretende así lograr una plataforma experimental, versátil y de fácil de utilizar, por un coste económico considerablemente reducido.

2.4. Sistema de control

Hay varios tipos de equipos que podrían servir como dispositivo de control, sistema informático, en una nariz electrónica, desde chips microcontroladores a ordenadores, pasando por plataformas embebidas. Dado que uno de los requisitos del proyecto es lograr una plataforma de tamaño reducido, para que sea de fácil transportar, se descarta de partida el uso de un ordenador como controlador del sistema.

Antes de profundizar, es conveniente entender las diferencias entre un microcontrolador y una plataforma embebida, así como las ventajas o inconvenientes de una frente a la otra.

- **Microcontrolador:** se tratan de un circuito integrado programable en cuyo interior contienen una unidad central de procesamiento (CPU), unidades de memoria (RAM o ROM), puertos E/S y periféricos. Todos conectados entre sí, permitiendo ejecutar las ordenes grabadas en la memoria. Un microcontrolador contiene todo lo necesario para el control de un sistema externo, y nada más. Esta limitación de la funcionalidad del microcontrolador a los requisitos básicos de una unidad de control hace que su implementación sea económica y sencilla. La antigua plataforma del GNB [1] utilizaba un PIC.

- **Plataformas embebidas:** por sistema embebido, o también denominado empotrado, se reconoce un sistema de computación diseñado y empleado para cubrir una serie de necesidades específicas. La parte fundamental de estos sistemas es la placa base (plataforma o tarjeta embebida) que contiene el microprocesador, las memorias, los puertos de comunicación (conectores físicos), otros chips, etc.

La principal diferencia entre un procesador embebido y un microcontrolador se encuentra en sus componentes y la integración de los mismos. Por un lado, un procesador embebido controla el sistema del que forma parte, pero requiere recursos externos tales como memoria. En un microcontrolador se encuentra todo lo necesario para controlar un sistema en un solo chip, es decir, podría contener un procesador embebido como parte de sus componentes, que puede combinar con otras piezas de la computadora, tales como la memoria o los registros de señal.

Una plataforma embebida a diferencia de los microcontroladores, viene con el hardware auxiliar necesario, y a diferencia de los ordenadores, aporta un consumo menor y unas prestaciones más que suficientes para este trabajo. Además, actualmente existen sistemas embebidos que presentan sistemas operativos potentes como Linux o Windows, o versiones reducidas de los mismos. Y, por consiguiente, las herramientas de desarrollo software típicas de estos sistemas operativos.

Por todo esto, se ha decidido que es el elemento adecuado para el desarrollo del proyecto. En el anexo [A] se puede ver más información acerca de los sistemas embebidos.

Actualmente, el mercado de los sistemas embebidos ofrece muchas alternativas, aunque las tres más populares son: Arduino, Raspberry Pi y BeagleBone Black.

3. Estudio y elección de componentes de la plataforma

3.1. Introducción

Como se ha comentado anteriormente, una nariz artificial es un equipo de medida formado por una serie de módulos que trabajan de forma conjunta con el objetivo de cuantificar y/o clasificar muestras gaseosas o aromáticas. Una EN's tiene como componentes los siguientes elementos:

- **Módulo de muestreo**. Donde se realiza la adquisición de la muestra de gas. Estará compuesto por tres matraces con distintas sustancias químicas líquidas que desprendan olor y circuito de circulación de gases cuyo fin es hacer llegar la muestra al sensor.

- **Matriz de sensores químicos**. Donde se lleva a cabo la adquisición de información proveniente del odorante. El sensor varía sus características eléctricas en función de la pluma de odorante detectada.

- **Sistema informático.** Se encargará de monitorizar y grabar los datos que se obtienen de las mediciones. Es decir, es un sistema de adquisición y procesado. Se utilizará una plataforma embebida. Además, servirá para controlar el flujo de odorante, temperatura de calentamiento del sensor, tipo de modulación a implementar, etc.

- **Circuitos electrónicos.** Módulos que permiten el acondicionamiento de las señales del sistema.

3.2. Placa microcontroladora

Como se indicó en el estado del arte, hay varios dispositivos capaces de controlar la plataforma de experimentación; pero dadas sus características de bajo consumo, la utilización de sistemas operativos como Linux y su bajo coste; se ha optado por la elección de un sistema embebido.

Actualmente en el mercado hay una amplia variedad de sistemas embebidos, siendo los más destacados los elaborados por las empresas Arduino, Raspberry y Beaglebone.

Arduino es una compañía italiana de hardware libre, que elabora placas de desarrollo que integran un microcontrolador y un entorno de desarrollo (IDE). En cuanto a hardware consiste en una placa de circuito impreso con un microcontrolador, puertos digitales y analógicos de entrada/salida. Dependiendo del modelo encontraremos diferencias en cuanto a procesadores más o menos potentes, mayor o menor número de puertos de entrada/salida, conectividad con hdmi, ethernet, etc. Además, hay una gran cantidad de placas de expansión para ampliar el funcionamiento de la placa arduino. En cuanto al software, consiste en un entorno de desarrollo propio con un lenguaje de programación basado en Wiring.

Por todo esto, se puede concluir que Arduino es un sistema idóneo para proyectos simples. Algunas de las placas Arduino más destacas son las expuestas en la siguiente imagen:

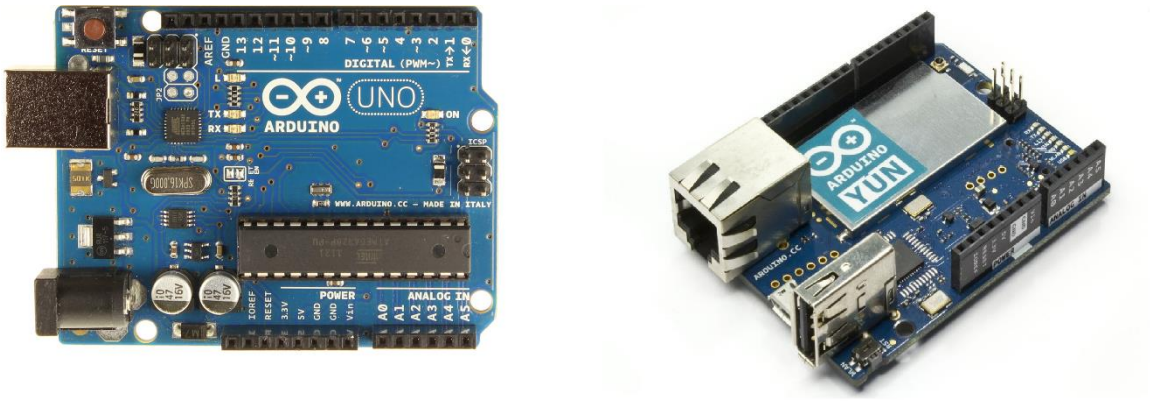


Figura 3. 1: *Arduino UNO (imagen izquierda). Arduino YUN (imagen derecha).*

Raspberry Pi se desarrolla en Reino Unido, y nació con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. En la actualidad, se pueden encontrar tres versiones de este sistema embebido (aunque al inicio de este proyecto soló se comercializaba Raspberry Pi model 1), para este proyecto se tuvo sólo encuentra la primera de las versiones comercializadas (se puede observar en la Figura 3. 2).

La placa Raspberry Pi cuenta con una versión de Debian, denominada RasBian, como sistema operativo, lo que la hace más versátil que las placas de Arduino, pero tiene un número muy limitado de entradas/salidas digitales y analógicas. Lo que la limita para interactuar con sensores u otros dispositivos externos. Sin embargo, destacan por sus entradas y salidas multimedia (salidas de video y audio, entrada de video). Es idónea para proyectos multimedia o en general proyectos complejos basados en Linux.

Figura 3. 2: *Raspberry Pi model 1.*

BeagleBone Black es una opción más adecuada para proyectos que son demasiado complejos para Arduino pero que no necesitan de complejos gráficos como Raspberry Pi. La plataforma de desarrollo BeagleBone Black permite una fácil conexión a internet y es idónea para emplear en proyectos con sensores externos. Esta tarjeta presenta prestaciones aún mejores que la Raspberry Pi en aspectos de capacidades de procesamientos, periféricos y GPIOs, sin embargo, una de las más grandes desventajas es que solo tiene un puerto USB Host.

Finalmente, se ha decidido el uso de BeagleBone Black (a partir de ahora para hacer referencia a la misma se utilizará BBB) porque dispone de una capacidad computacional alta, muchas posibilidades de comunicación con el exterior (gran variedad de pines de comunicación de entrada/salida), además de su bajo coste.

3.2.1. BBB. Características Generales

Las placas BeagleBone son una familia de microcomputadores de hardware libre que destacan por su baja potencia, sistema operativo embebido basado en Linux y que están enfocadas a su utilización con código abierto (open-source). Útiles en entornos educativos, de experimentación, investigación y proyectos de bajo coste.

Para el presente proyecto se ha decidido utilizar la última versión de la misma, la denominada como BeagleBone Black, cuyas características se exponen en la siguiente tabla.

	CARACTERÍSTICAS	
Procesador	Sitara AM3358BZCZ100 1GHz, 2000 MIPS	
Motor gráfico	SGX530 3D, 20M Polygons/S	
Memoria SDRAM	512MB DDR3L 800MHz	
Onboard Flash	4GB, 8bit Embedded MMC	
PMIC	TPS65217C PMIC regulator and one additional LDO	
Soporte Debug	Optional Onboard 20-pin CTI JTAG, Serial Header	
Fuente alimentación	miniUSB or DC Jack	5VDC External Via Expansion Header
PCB	3,4" x 2.1"	6 layers
Indicadores	1-Power, 2-Ethernet, 4-User Controllable LEDs	
HS USB 2,0 Client Port	Access to USB0, Client mode via miniUSB	
HS USB 2,0 Host Port	Access to USB1, Type A Socket, 500mA LS/FS/HS	
Puerto Serie	UART0 access via 6 pin 3.3V TTL Header. Header is populated	
Ethernet	10/100, RJ45	
SD(MMC Conector	microSD, 3.3V	
Entradas de Usuario	Reset button Boot button Power button	
Salida video	16b HDMI, 1280x1024 (MAX) 1024x768, 1280x720, 1440x900, 1920x1080@24Hz w/EDID Support	
Audio	Via HDMI Interface, Stereo	
Conectores de expansión	Power 5V, 3.3V, VDD_ADC(1.8V) 3.3V I/O on all signals McASP0, SPI1, I2C, GPIO(69 max), LCD, GPMC, MMC1, MMC2, 7 AIN(1.8V MAX), 4 Timers, 4 Serial ports, CAN0, EHRPWM(0.2), XDMA Interrupt, Power Button, Expansion Board ID (Up to 4 can be stacked)	
Peso	1.4 oz (39.68grams)	

Tabla 2: *Características generales de la placa microcontroladora BeagleBone Black.*

En la Figura 3. 3 se observa el diagrama de bloques de la plataforma de desarrollo BBB. A continuación, se describen las características y los principales elementos del diseño de la tarjeta BeagleBone Black (ver Figura 3. 4 y Figura 3. 5):

- Procesador: utiliza un microprocesador AM3358 en un encapsulado de 15x15. Su velocidad real viene determinada por cómo se la conexión y la manera de su alimentación.

- Memoria: Usa una memoria DDR3 RAM 4Gb (Double Data Rate type three RAM) de 16 bits, con configuración de 512MB a 800MHz. Tiene una EPROM de 4KB con información de la placa. Además, tiene una memoria interna embebida (eMMC) de 4GB.

- Control de alimentación: El circuito integrado TPS65217C está diseñado para dar soporte a los procesadores AM335X. Se usa junto con un regulador LDO externo para alimentar todo el sistema.

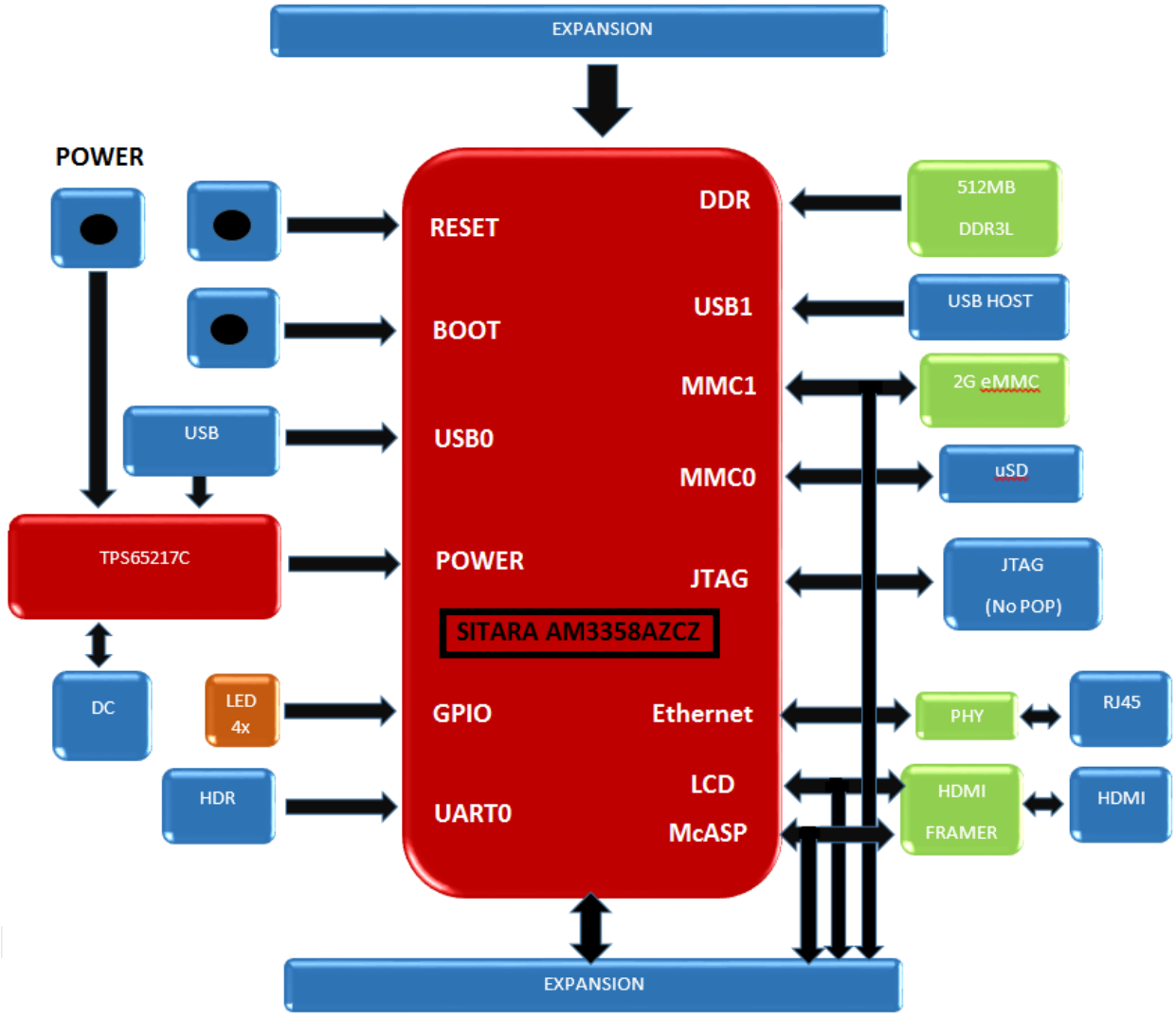


Figura 3. 3: *Diagrama de bloques del sistema BBB. Imagen replicada de [15].*

- Conector microSD.

- Interfaz USB: Tiene un USB HUB que permite concentrar los dos puertos USB para operar como: comunicación serie por puerto USB, JTAG por puerto USB y acceso directo al procesador por el puerto USB (puerto USB0).

- Puerto USB1: La tarjeta dispone de un conector USB tipo A que conecta con USB1 del procesador. El puerto puede proporcionar alimentación de encendido (5 voltios) y hasta 500mA dependiendo de cómo esté conectada a la alimentación.

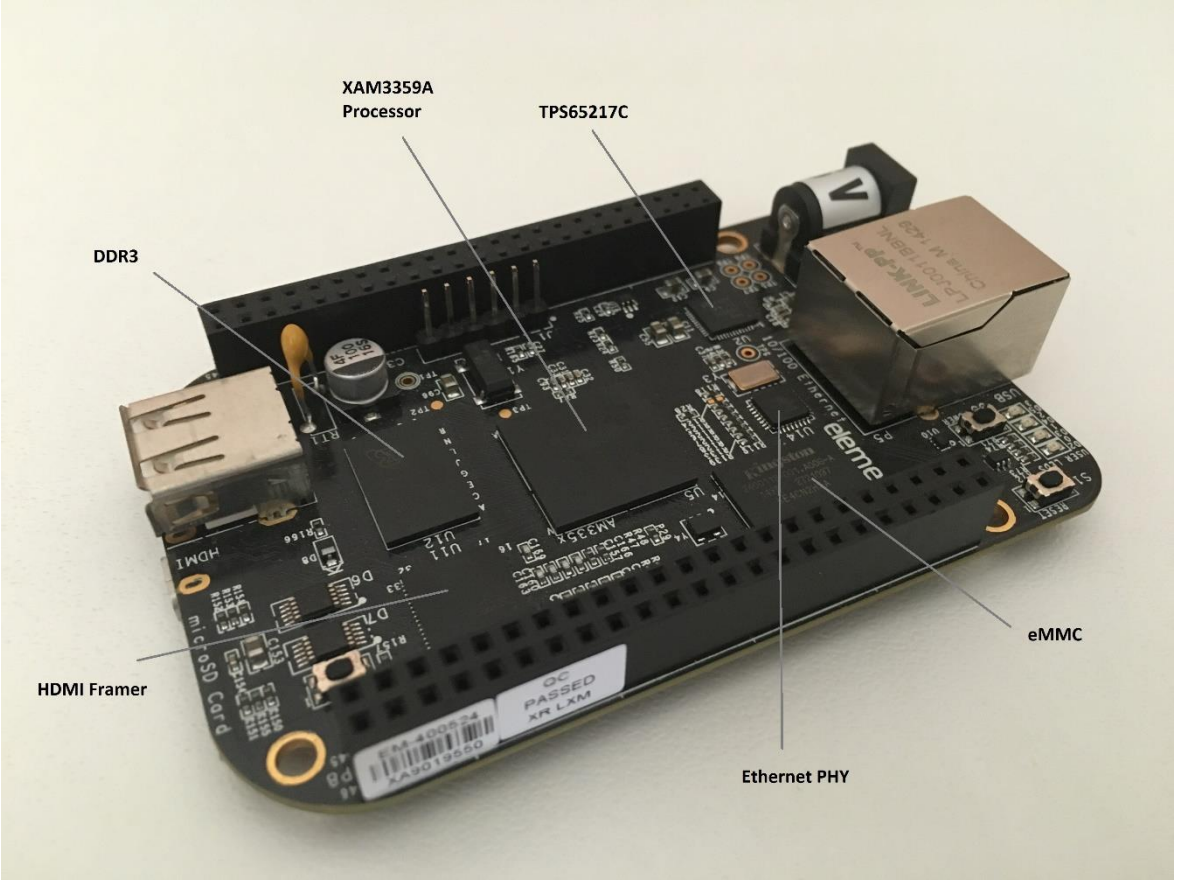


Figura 3. 4: *Componentes de la tarjeta BBB.*

- Puerto USB Cliente: Proporciona acceso a USB0 a través de USB HUB. Se mostrará en un PC como un dispositivo estándar.

- Alimentación: puede ser alimentada a través de un puerto USB de un ordenador o desde una fuente de alimentación externa de 5 VDC. Si se alimenta por puerto USB el procesador ve mermada su velocidad.

- Botón Reset.

- Botón Apagado/Encendido.

- Indicadores: La tarjeta dispone de cinco LEDs, de los cuales cuatro pueden ser controlados por el usurario, el otro, muestra si la placa está siendo alimentada.

- Interfaces de Expansión: Dispone de dos conectores de 46 pines cada uno, para el acceso a las señales de expansión. Estos pines tienen distintas funciones que se explican en la sección [3.2.2].

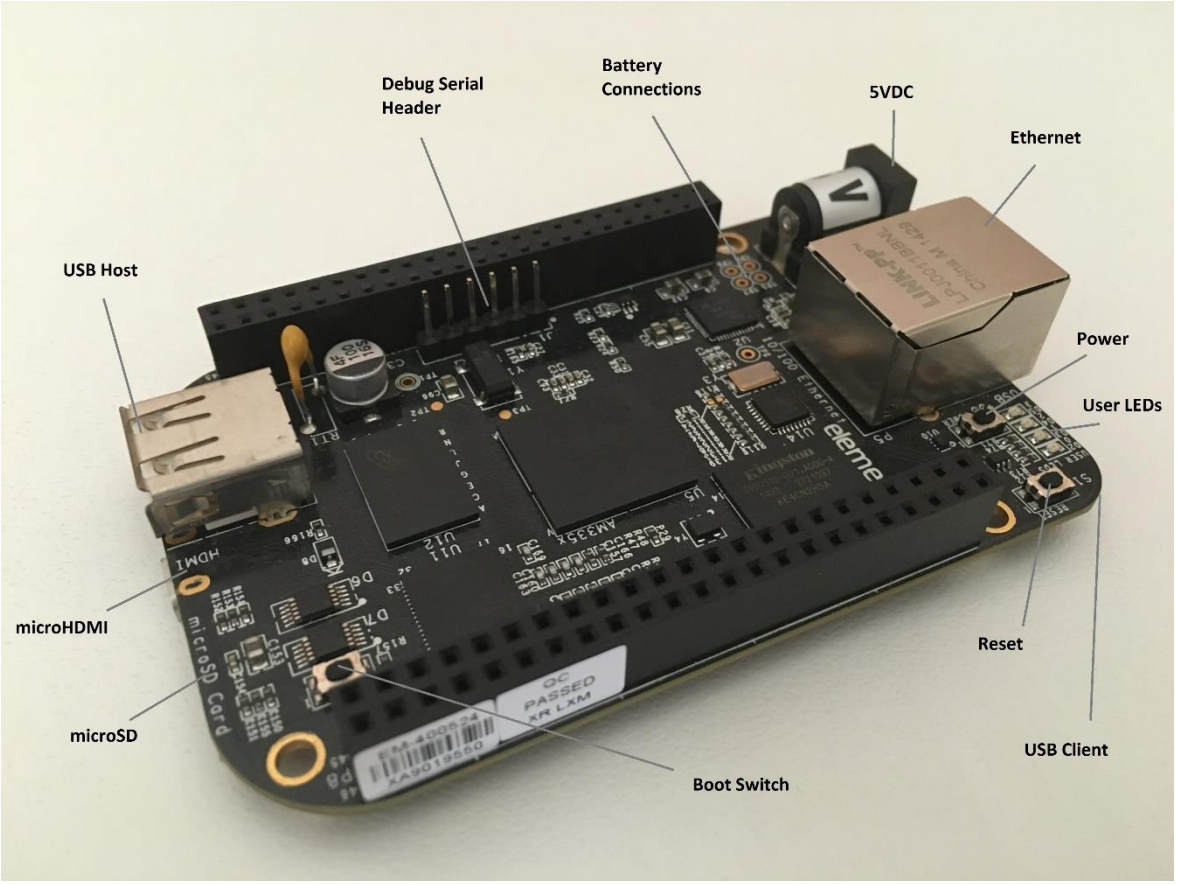


Figura 3. 5: *Conectores e interruptores de la tarjeta BBB.*

3.2.2. BBB. Pines de expansión entrada/salida

Los pines de expansión permiten conectar otros sistemas a la placa como periféricos, pudiendo controlar o recibir información de ellos. Se trata de un total de 92 pines dispuestos en dos módulos P8 y P9, se pueden apreciar en la Figura 3. 6.

En el anexo **[**B**]**, se puede encontrar el mapeo de los modos en los que funciona cada uno de los pines. A continuación, se enumeran los distintos modos de funcionamiento y los periféricos adicionales que se pueden conectar a la placa.

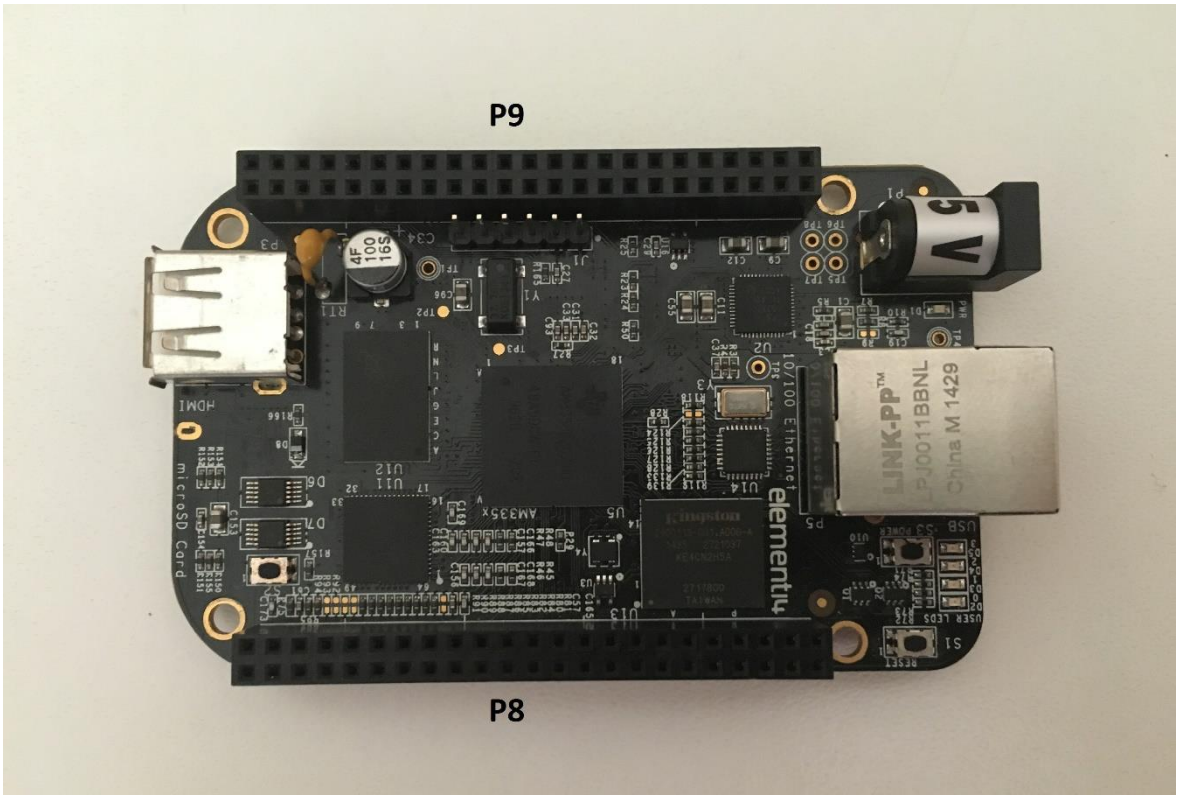


Figura 3. 6: *Pines expansión BeagleBone Black*

- LCD**.** Es posible conectar una placa LCD de 24 o 16 bits a la BBB.

- GPMC**.** Los puertos de expansión disponen de líneas GPMC.

- MMC1. Permite acceder a una tarjeta MMC (MultiMediaCard) localizada en el conector microSD.

- SPI. En los conectores de expansión se dispone de dos puertos de SPI (Serial Peripheral Interface) disponibles, SPI0 y SPI1.

- Puerto Serial. Hay cuatro puertos serie en los conectores de expansión (UART1, UART2, UART3, UART4 y UART5). Los dos primeros tienen señales Rx y Tx, el UART3 tiene RTS y CTS, y los dos últimos tienen todas.

- Conversores Analógicos-Digitales: Existen siete ADC con un voltaje de conversión máximo de 1.8 voltios. Los pines de entrada analógicos aceptan hasta 1,8 voltios y tienen una resolución de 1 milivoltio. Tienen una resolución de 12 bits.

- GPIO. Dispone de 65 pines de General Purpose Input/Output. Estos pines son de 3,3 voltios, lo que correspondería a una 1 digital.

- CAN Bus. Tiene dos interfaces CAN bus.

- TIMERS. Posee cuatro temporizadores de salida.

- PWM. Dispone de ocho salidas PWM (Pulse Width Modulation). Seis de ellas de alta resolución y dos de eCAP. La técnica PWM permite variar una señal de salida digital a alta frecuencia. El resultado trata de simular una señal analógica mediante la variación del ciclo de subida en la señal periódica transmitida (véase Figura 3. 7).

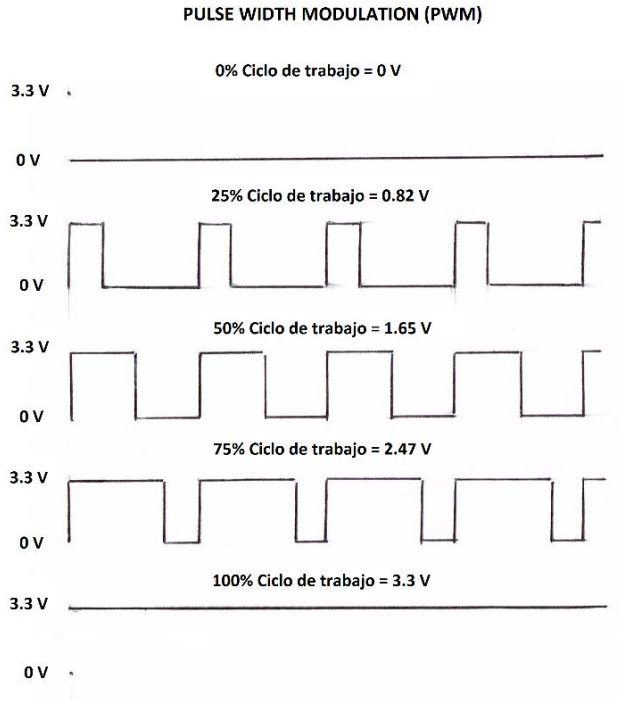


Figura 3. 7. *Pulse Width Modulation.*

- Alimentación externa. Posee ocho pines de GND (tierra), dos pines de alimentación de 3.3 voltios y cuatro de 5 voltios, estos último sólo funcionan cuando la placa está alimentada por transformador.

Toda la información de la placa BeagleBone Black ha sido obtenida de su manual, que puede verse en [15].

Tras enumerar las características de la placa, se puede decir que las principales ventajas que han llevado a la elección de la BeagleBone Black, en contra de Arduino y Rapsberry, como placa controladora de la plataforma de detección son:

- Procesador más potente en velocidad y capacidades Cortex-A8.

- Memoria interna para el sistema operativo de 4GB incluida.

- Más entradas y salidas.

- Botones de encendido y reset.
- Muy bajo consumo.
- Posibilidad de programarla al estilo Arduino con scripts.
- Por 55€ incluye la memoria y el cable USB listo para usarla.
- Fuente de alimentación de 5V a 1A opcional por 15€.

3.2.3. Limitaciones de la placa Beaglebone Black

Adquiere especial importancia este punto para realizar un correcto acondicionamiento de señales que entren o salgan por los pines I/O de la BBB. Se analiza el manual de referencia y se obtienen los siguientes datos límite para los modos a utilizar en el proyecto:

I/O	Nombre	Nº PINES	Rango Voltaje	Corriente max	Tiempo muestreo
ANALOG (I)	AIN_X	7	0 - 1,8 (V)	2 (μA)	125 (ηs)
DIGITAL (I)	GPIO_X	66	0 o 3,3 (V)	4 -6 (mA)	140 (ηs)
DIGITAL (O)	GPIO_X	66	0 o 3,3 (V)	4 - 6 (mA)	95 - 105 (ηs)

Tabla 3: *Limitaciones impuestas por la placa BBB.*

Los rangos indicados en la Tabla 3, especialmente para los pines de entrada, no pueden ser excedidos o se corre el riesgo de quemar la placa.

Los puertos analógicos (AIN_X) tienen como límite de entrada un voltaje de 1.8 voltios, que no podrá ser excedido, y soporta una corriente máxima de entrada de 2 μA. Dichos pines se utilizarán para la lectura de datos provenientes del sensor, por lo que habrá que realizar un circuito de acondicionamiento para garantizar que no se sobre pasen los límites impuestos por la BBB. La placa cuenta con dos pines de referencia para estos puertos, el VDD_ADC que entrega 1.8 voltios y el GND_ADC que es la referencia a tierra.

Para el caso de las entradas digitales (GPIO_X) ofrecen un rango dinámico de 0 a 3,3 voltios que no podrá tampoco ser sobrepasado.

Por lo cual será necesario crear circuitos para la protección de la placa, que se explicarán en la sección [5.2].

3.3. Sensor quimiorresistivo. TGS 2600

Los sensores Figaro TGS (Taguchi Gas Sensor) pertenecen al tipo de sensores de óxido metálico de capa gruesa. Su coste es muy bajo porque se producen industrialmente. Esto permite también poder adquirirlos fácilmente en cualquier parte del mundo. Por otra parte, presenta una larga vida útil, y ofrecen una sensibilidad considerable a la presencia de gases. A todo esto, hay que añadir que la resistencia del sensor y la concentración de gases reductores ante los que se haya presente se puede expresar usando la ecuación descrita al final de la sección [2.2.1]:

$$R_s = A[C]^{-\propto}$$

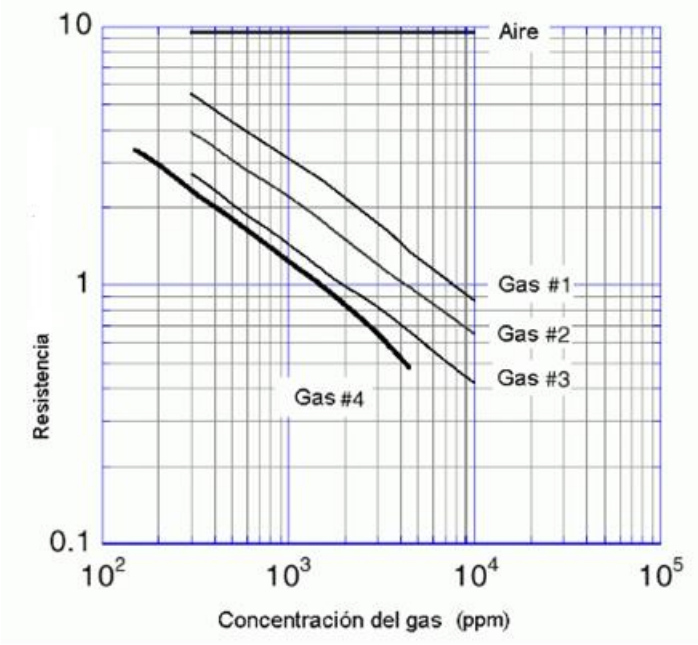


Figura 3. 8: *Ejemplo de variación de la Resistencia del sensor en función de la concentración de gas para sensores TGS de Figaro.*

Tal y como se muestra en la Figura 3. 8, la relación entre la resistencia y la presión parcial de los gases a analizar es lineal dentro de una escala logarítmica. La figura es un ejemplo genérico del comportamiento de los sensores TGS de la casa Figaro. Cada uno de los sensores Figaro reaccionan ante varios gases, aunque existen ciertas especias ante las que se muestran especialmente sensibles. Esta sensibilidad va a depender también, en gran medida, de la temperatura de operación. Como las resistencias internas de los sensores varían en función del gas ante el que se encuentren presentes, la sensibilidad típica de cada uno de

ellos viene expresada por la relación entre su resistencia ante varias concentraciones de gases (Rs) y ante cierta concentración de referencia (Ro).

Antes de introducir el sensor que se va a utilizar en el proyecto conviene hacer referencia a dos comportamientos típicos de este tipo de sensores:

- **Acción inicial**. Tal y como se ve en la Figura 3. 9 todos los sensores exhiben un comportamiento conocido como acción inicial cuando permanecen sin funcionar durante cierto periodo de tiempo. Cuando el sensor se conecta se produce una caída intensa en la resistencia que dura unos segundos, independientemente de que haya presencia de gas o no, aunque en el caso de que lo haya la caída es característica. Posteriormente alcanza el nivel adecuado a la atmósfera en la que se encuentre. La magnitud de esta acción inicial depende de las condiciones del ambiente durante el almacenamiento y de su duración, y es distinto en cada sensor.

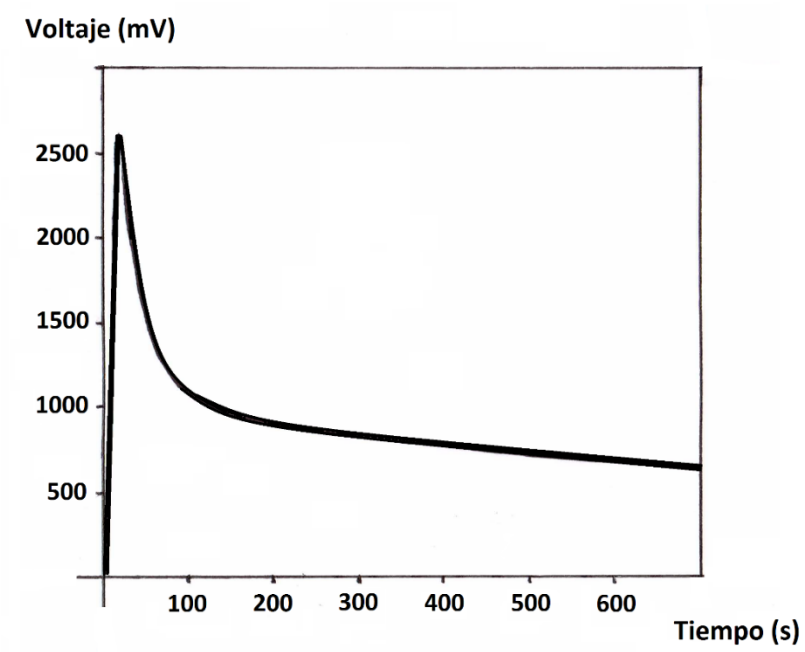


Figura 3. 9: *Acción inicial sensor TGS de Figaro.*

- **Respuesta transitoria**. En la Figura 3. 10puede observarse el comportamiento típico de un sensor cuando es expuesto a un gas. Primero la resistencia aumenta hasta alcanzar un valor estable, y cuando se retira el gas recupera su valor inicial después de un corto periodo de tiempo. La velocidad de la respuesta y la reversibilidad dependerá del modelo del sensor y del gas objeto de estudio.

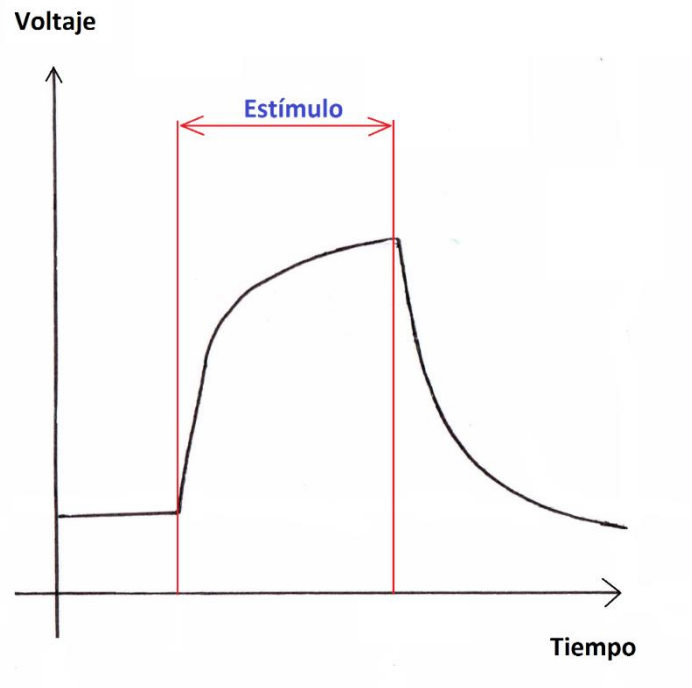


Figura 3. 10: *Respuesta transitoria sensor TGS de Figaro.*

El sensor utilizado para captar partículas en este proyecto es un TGS2600. Especializado en detección en el aire de partículas contaminantes. Se utiliza normalmente en aplicaciones tales como control de calidad del aire, control de ventilación y limpieza de aire.

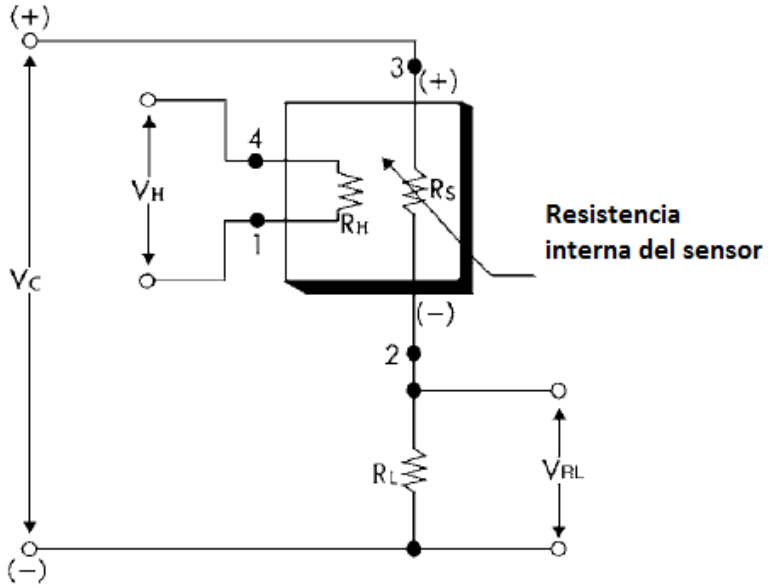


Figura 3. 11: *Circuito de acondicionamiento del sensor TGS2600. Imagen adaptada del dataasheet del TGS2600 [10].*

El elemento sensor está compuesto por una capa de semiconductor de óxido metálico formado por sustrato de aluminio. En la presencia de un gas detectable, la conductividad del

sensor aumenta dependiendo de la concentración del gas en el aire. Un sencillo circuito convierte el cambio de la conductividad en una señal de salida que corresponde a la concentración de gas (véase Figura 3. 11).

Atendiendo al circuito eléctrico, el sensor requiere de dos entradas de voltaje: voltaje de calefacción (V_H) y voltaje de circuito (V_C). El voltaje de calefacción es aplicado al calefactor integrado para así mantener al elemento sensor en una temperatura determinada, la cual es óptima para poder medir. El voltaje de circuito es aplicado para así poder medir el voltaje (V_{OUT}) a través de una resistencia de carga (R_L), la cual está conectada en serie con el sensor. El valor de la resistencia de carga debe ser escogido para optimizar el valor del umbral de alarma, manteniendo el consumo de energía en el semiconductor por debajo de 15 mW. El consumo de energía (P_S) será máximo cuando el valor de RS sea igual al de R_L durante la exposición al gas.

Como se puede observar en la Figura 3. 11, el circuito de medición que se utiliza se basa en el principio de divisor de tensión. Del análisis del circuito de la figura se desprende la siguiente información:

El consumo de energía viene dado por: $P_s = \frac{(V_c - V_{out})^2}{R_s}$

Resistencia sensor vienen dada por: $R_s = \frac{V_c \times R_L}{V_{out}} - R_L$

Dónde: R_s, resistencia eléctrica del sensor
Vc, tensión de referencia
Vh, tensión de calentamiento
Vout, tensión de salida (tensión a medir en la resistencia R_L)
R_L, Resistencia de carga

Las principales especificaciones técnicas del TGS2600 se exponen en el siguiente cuadro:

Especificaciones TGS2600			
Vh	Heater voltaje	5.0 ± 0.2 (V) DC/AC	
Vc	Circuit voltaje	5.0 ± 0.2 (V) DC/AC	Ps ≤ 15mW
RL	Load resistance	0,45 KΩ min	
Rh	Heater resistance	approx. 83Ω	
Rs	Sensor resistance	10K - 90KΩ in air	

Tabla 4: *Especificaciones TGS2600. Tabla generada a partir de su datasheet [10].*

3.3.1. Características de los sensores TGS2600

3.3.1.1. Sensibilidad

La sensibilidad de estos sensores está definida por la relación entre los cambios de concentración del gas y las variaciones de la resistencia del sensor, basada en una función logarítmica. Cada tipo de sensor tiene su propia sensibilidad característica, la cual le permite ser empleado para diferentes propósitos. La siguiente figura presenta la gráfica de sensibilidad del TGS2600.

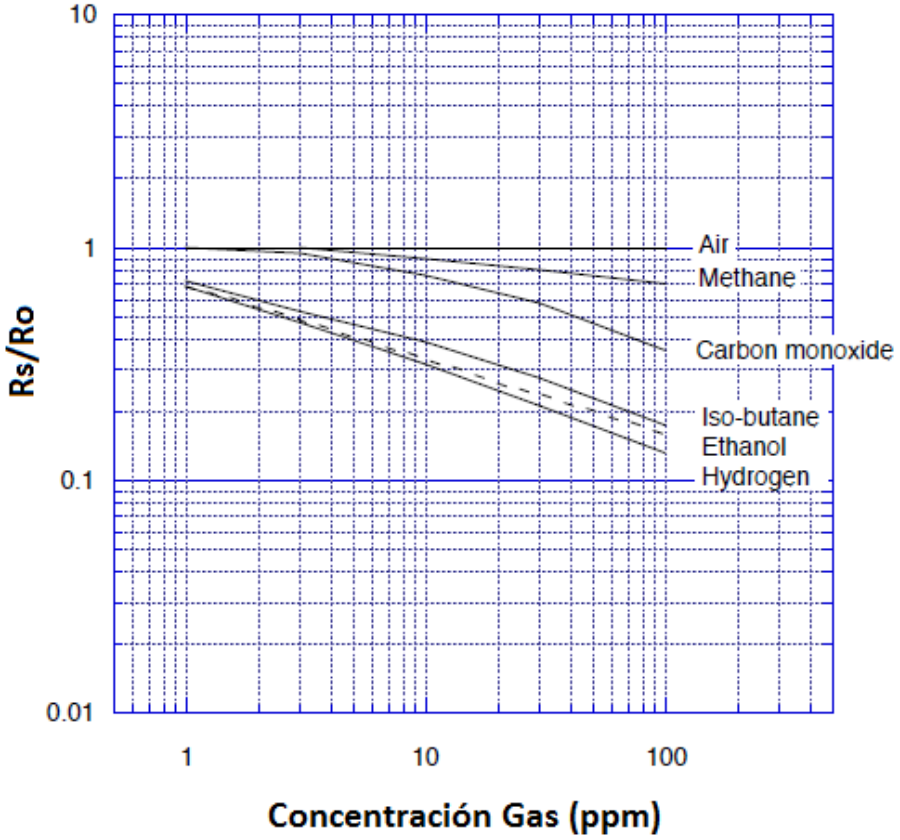


Figura 3. 12: *Sensibilidad característica del TGS2600. Imagen adaptada de datasheet del TGS2600 [10],*

En el eje de ordenadas se representa el valor de las resistencias del sensor medida para varias concentraciones de gases, comparada con la resistencia del sensor en aire libre.

3.3.1.2. Efectos de la temperatura y humedad.

Como la detección en los TGS se debe a la adsorción y desorción química de gases en la superficie del sensor, la temperatura y la humedad afectan directamente a la sensibilidad, ya

que provocarán un cambio en la magnitud de las reacciones superficiales. Tal y como se puede comprobar en la Figura 3. 13, donde se ve que su variación provoca un cambio notable en la resistencia del sensor.

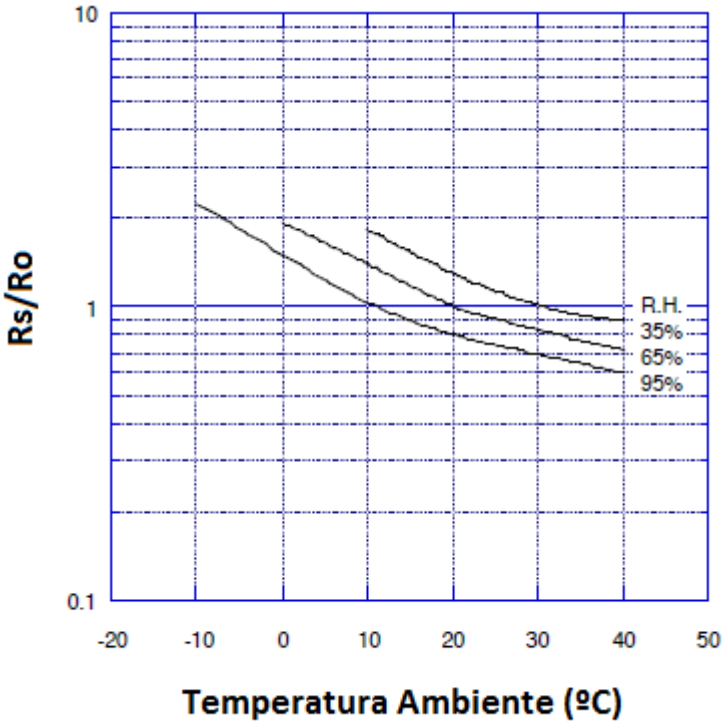


Figura 3. 13: *Influencia de la temperatura sobre la resistencia del sensor TGS2600. Donde Rs es la resistencia del sensor en aire libre para varias temperaturas/humedades y Ro la resistencia del sensor en aire libre a 20ºC y 65% de humedad relativa. Imagen obtenida del datasheet del TGS2600 [10].*

Se puede provocar un aumento de la temperatura si se incrementa la tensión de heater del sensor, por lo que es importante controlar esta variable. Es recomendable utilizar con estos sensores circuitos que compensen los efectos de la temperatura.

En cuanto a la humedad, las moléculas de agua adsorbidas por la superficie del sensor le afectan disminuyendo su resistencia, es decir, a medida que aumenta la humedad, el valor de la resistencia del sensor va disminuyendo. Cuando pretende detectar un gas en presencia de un ambiente húmedo, la detección resulta más costosa. Por lo tanto, se debe eliminar esta dependencia, para ello existen diversas técnicas:

- Midiendo la humedad y realizando su compensación por software.
- Ciclo térmico pulsado (actuando sobre la resistencia de heater)
- A través de filtros mecánicos/químicos que eliminan la humedad
- Manteniendo la humedad constante.

Será un aspecto a tener en cuenta en el proyecto, pues variaciones bruscas en la temperatura o la humedad ocasionarán falsas detecciones de odorante. Por ello se realizará un circuito de monitorización de humedad y temperatura que se integrará en la plataforma.

3.3.1.3. Recomendaciones de uso sensores TGS.

A continuación, se presentan una serie de recomendaciones, para hacer un buen uso de los sensores de óxido metálico de Figaro:

1. Se recomienda utilizar los circuitos de medida mostrados en las hojas de especificaciones ofrecidas por el fabricante.

2. La humedad y la temperatura afectan la repetitividad de los sensores. Si incluimos un termistor para compensar dicha dependencia, disminuiremos sus derivas, consiguiendo fijar el punto de trabajo del sensor.

3. Para calibrarlos se debe llevar a cabo bajo unas condiciones de temperatura y humedad controladas, y usando aire limpio y un gas puro. Se recomienda unos 20ºC ± 2ºC, 65 ± 5% R.H.

4. Los sensores TGS pueden ser empleados con polarización continua o alterna. Sin embargo, la resistencia del sensor puede cambiar de valor cuando es polarizado inversamente.

5. Los sensores antes de ser usados deben estar, como mínimo, durante una semana quemando (preheating time) sin ser expuestos a ningún contaminante, con la finalidad de que alcancen un valor de resistencia estable y eliminar cualquier impureza de su interior.

3.4. Control temperatura por modulación de amplitud

En la obtención de información con sensores quimioresistivos adquiere un papel fundamental la temperatura en la que se emplaza el sensor. Dependiendo de esta temperatura la salida del sensor será distinta ante la presencia de una concentración determinada de odorante, dando lugar un amplio abanico de modulaciones sobre la amplitud de la señal temperatura.

En trabajos anteriores del GNB se han usado diferentes tipos de modulación de temperatura para la consecución de diferentes estudios. El control mediante la modulación de temperatura nos facilita una fuente de información en los valores transitorios del sistema, lo que nos permite controlar la histéresis del sensor y modificar la sensibilidad del sistema.

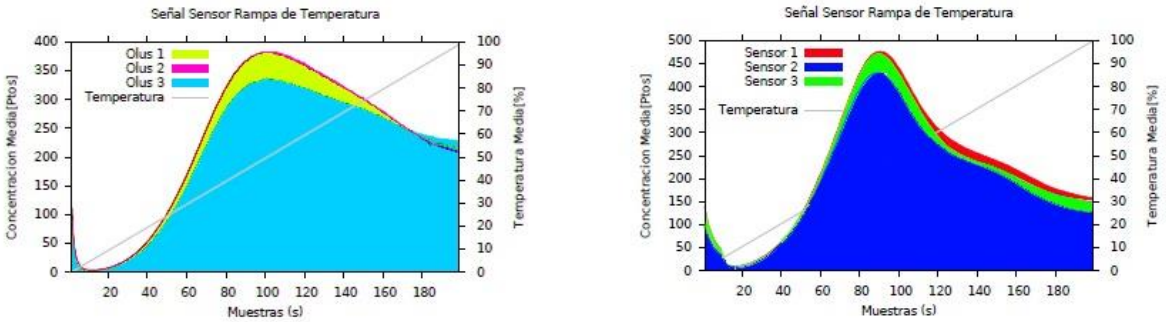
La modulación de temperatura se puede realizar de diversas maneras, básicamente, se utiliza la salida del sensor como la señal de entrada del calentador, tras una modulación previa. La diferencia radica en la forma de utilizar esta señal de salida de sensor, es decir, en la modulación que se utilice para su acondicionamiento. Se puede realizar una señal de forma cuadrada [19], se puede estipular un crecimiento lineal [20], a través de impulsos de temperatura [21] o con el uso de señales sinusoidales.

Para la realización de este tipo de modulación se partirá del sistema de acondicionamiento realizado en el trabajo final de carrera: '*Uso de una nariz electrónica ultra-portátil en robots para la detección de fuentes de odorante*' [7]. En dicho trabajo se realiza un robot capaz de localizar una fuente de odorante, este robot contiene cuatro sistemas de narices electrónicas, basadas en un sensor TGS2611 [35], un circuito de acondicionamiento y una modulación de temperatura por amplitud. Se mejorará el circuito de adaptamiento y se modificará para su uso con un TGS2600.

El sensor utilizado en este proyecto, aunque de la misma familia (Figaro), será distinto. Por lo que se realizará, en la sección [5], un breve estudio y comparación de ambos, para modificar el acondicionamiento de la forma adecuada.

Modulación por rampa de temperatura:

Se trata de realizar un incremento lineal de la temperatura de calentamiento del sensor desde su valor mínimo al máximo. A la vista de la Figura 3. 14, este incremento lineal provoca un reinicio de los valores de sensibilidad del sistema y se aprecia que el efecto de histéresis es prácticamente inapreciable. Este comportamiento ha sido utilizado antes en [2][7] donde se calentaba el sensor hasta el 100% de su temperatura para limpiar el sensor de posibles impurezas.



(a) Señal Rampa Sistema Olus. (b) Señal Rampa Sistema Integrado.

Figura 3. 14: *Incremento lineal de temperatura. Imagen izquierda, Sistema Olus [2]. Imagen modificada de [7]. Imagen derecha, Sistema TGS2600. Gráficas modificadas de [7].*

Observando las imágenes anteriores, se puede establecer de manera aproximada el margen dinámico de temperaturas sobre el cual la sensibilidad del sensor es más o menos

lineal. En la se muestra una prueba realizad para el sensor TGS2600, en ella se observa que su comportamiento es el mismo que el de los sensores TGS2611 de [7].

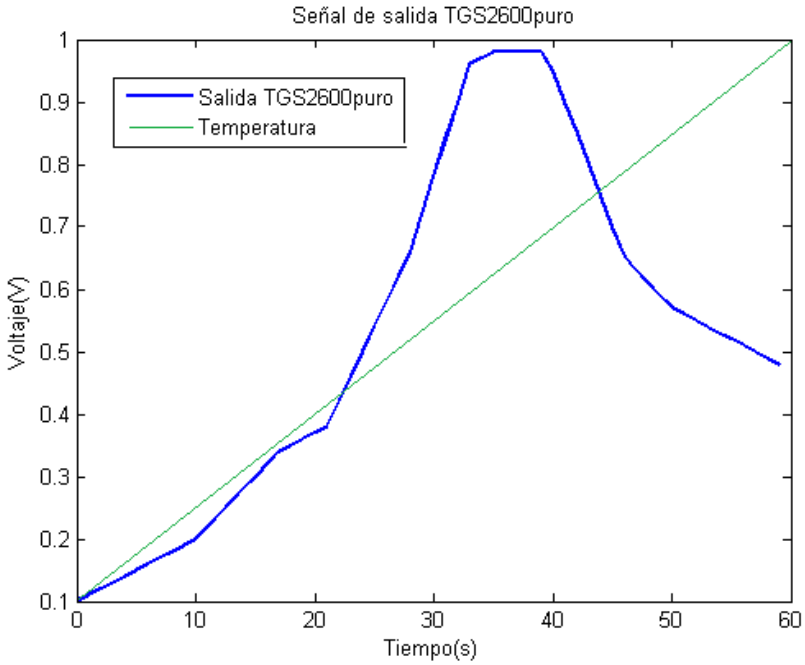


Figura 3. 15: *Respuesta del sensor TGS2600 con incremento lineal de temperatura. Gráfica sacada con la plataforma.*

En el presente proyecto se utilizará la modulación por rampa de temperatura como un paso previo a cada experimentación, con el fin de realizar una correcta limpieza del sensor. De forma que cada experimentación sea independiente de la exposición anterior del sistema.

Modulación sinusoidal:

Trata de hacer que la temperatura de calentamiento varíe de forma sinusoidal. Veremos que influencia tiene la oscilación de la temperatura sobre las señales de concentración.

Cierto es que cada sensor posee su propia curva característica de sensibilidad, pero como se aprecia en la Figura 3. 16, pequeñas variaciones de temperatura de calefacción son seguidas linealmente por todos los sensores tipo MOS con los que se han trabajado en el GNB.

Las siguientes figuras muestran las señales de concentración en aire libre, para distintos sensores de tipo MOS, con modulación sinusoidal de temperatura (izquierda) y con temperatura constante.

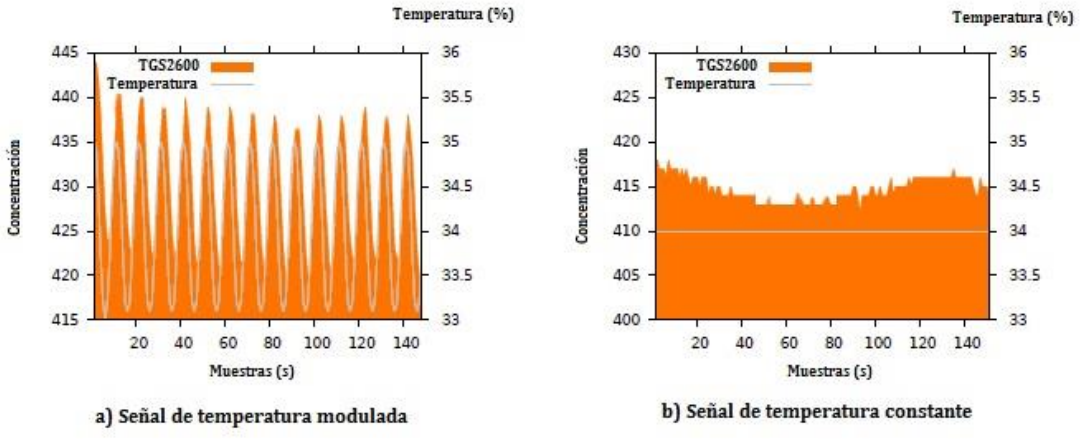


Figura 3. 16: *Efecto de la modulación sinusoidal de temperatura, gráfica izquierda. Frente la respuesta del sensor con temperatura constante, gráfica derecha. Imagen modificada de [7].*

Se observan dos claras ventajas de este tipo de modulación:

- Correlación de la señal obtenida de los sensores con la señal de temperatura. Se aprecia que la señal sigue forma sinusoidal. Esto otorga facilidad en el tratamiento y uso de las señales.
- Los picos de ruido que se aprecian en la señal sin modulación desaparecen en la señal con modulación sinusoidal de temperatura. Al aplicar la modulación se obtiene una señal de concentración del sensor con variación controlada.

Cobra vital importancia en este punto, la frecuencia de muestreo de la señal de salida. Pues si la frecuencia fuera muy alta el sistema no sería capaz de responder a las temperaturas impuestas y se perdería información. Por el contrario, si la frecuencia es muy baja, se obtendría información redundante y la información sería ineficiente.

En un sistema de medición, la frecuencia se calcula como la inversa del tiempo que se espera entre la adquisición de dos muestras consecutivas. Este tiempo tiene que ser lo suficiente como para que se pueda fijar una nueva temperatura de calentamiento.

Haciendo uso de la figura anterior, se observa claramente que para una frecuencia baja (f=2 muestras/segundo) se pierde mucha información. Mientras que para las frecuencias de 0.6 y 1 muestras/segundo el resultado es una señal con apariencia sinusoidal y estable. Entre estas dos últimas frecuencias, se elige como optima la de 1 muestra/segundo, pues el periodo entre muestras es menor que para la frecuencia de 0.6, y la resolución que se obtiene ya es suficiente.

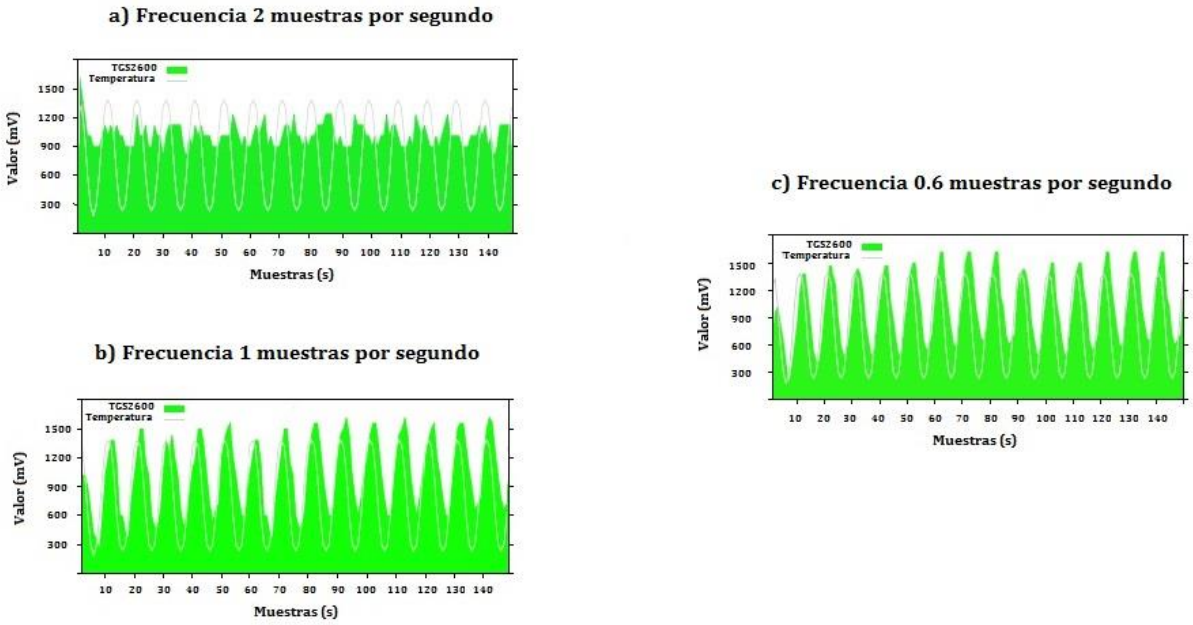


Figura 3. 17: *Comparativa entre frecuencias de muestreo. Sensor con temperatura de calentamiento modulada sinusoidalmente. Gráficas modificadas de [7].*

Modulación por regresión de temperatura:

En este caso la temperatura de calentamiento será autoadaptada en función de la salida anterior obtenida. Es decir, se realizará un circuito cerrado en el que se usa la salida del sensor como parámetro para calcular la siguiente temperatura de calentamiento.

De esta manera se calculará la nueva temperatura después de la captura de la salida de la tensión. Se aplicará un cálculo por regresión lineal de la tendencia en función de las medidas previas de odorante, que se denominará slope, y junto con una variable fija que denominaremos como tendencia de la modulación, se definirá la nueva temperatura de calentamiento como:

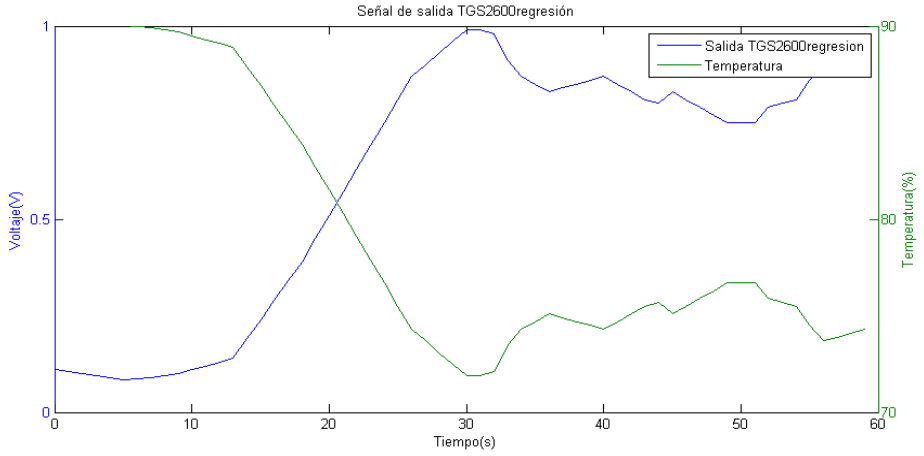


Figura 3. 18: *Gráfica de la señal de salida del algoritmo de regresión de temeperatura. En azul la señal medida y en verde la temperatura de calentamiento en cada instante de captura.*

$$T_{HEAT_{NEW}} = T_{HEAT_{OLD}-(SLOPE*TENDENCIA)}$$

En la Figura 3. 18 se observa el funcionamiento de la modulación por regresión de temperatura. Se aprecia que al introducir un odorante (instante de captura 10), el voltaje de salida sube. El código al detectar una subida en la salida bajará el valor de temperatura de calentamiento (de acuerdo con la ecuación antes descrita), y viceversa, si lo que detecta es una bajada. De forma que al final la señal de salida sea lo más constante posible.

3.5. Control temperatura por modulación de frecuencia

Como se ha indicado anteriormente, los mecanismos de detección con sensores MOS dependen de la temperatura. Siendo esta un método viable para controlar la sensibilidad y selectividad del sensor. A diferencia de las técnicas indicadas en el apartado anterior (véase 3.4), ahora se propone una modulación auto-adaptativa de la temperatura en frecuencia. En la presente sección, se expondrán las bases de funcionamiento de esta técnica desarrollada por el Profesor Martinelli y que explican con mayor detalle en sus publicaciones [8][9].

El método implementa el concepto de auto-modulación de la temperatura (*self-adapted temperatura modulation*), que se basa en la evidencia de que la sensibilidad al gas del sensor depende de la temperatura de funcionamiento, y que cada gas tiene su propia sensibilidad a la temperatura. Por lo tanto, una evaluación completa del estado del sensor se puede lograr si se tiene en cuenta ambas variables.

El sistema a implementar, básicamente, trata de colocar el sensor con sus terminales en un circuito de bucle cerrado, y usar la señal de salida como la señal de entrada del calentador del sensor. La siguiente figura muestra el diagrama de bloques del sistema:

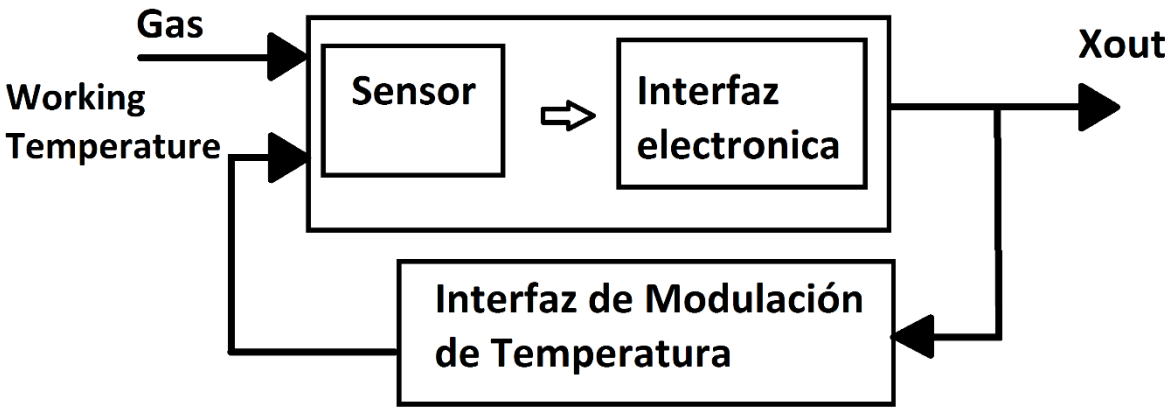


Figura 3. 19: *Diagrama bloques del Circuito cerrado que implementa The self-adapted termal modulation. Figura adaptada de [8].*

El concepto de modulación adaptativa de temperatura se compone de un sistema de circuito cerrado, en el que el primer bloque comprende el propio sensor junto con su circuito interfaz, mientras que el segundo, es el encargado de procesar la señal de salida del primer bloque (*Xout*), con el fin de obtener una señal adecuada de calentamiento del sensor entre unos rangos de voltaje y frecuencia definidos.

En la Figura 3. 19 se muestra la implementación en circuito del diagrama de bloques propuesto. En el que la resistencia del sensor es parte de un multivibrador astable. La elección del circuito oscilador se debe, principalmente, a la necesidad de simplificar la complejidad del circuito y la generación de una señal periódica que se usará para la interfaz de control de temperatura. El multivibrador astable es un circuito elemental caracterizado por una relación simple entre la salida en frecuencia de la señal y la resistencia del sensor.

El circuito oscilador se basa en el temporizador NE555 [17], un circuito integrado utilizado como generador de frecuencia de reloj en electrónica de consumo. En el esquema de la Figura 3. 20, el temporizador genera una señal de onda cuadrada, en el que la duración de los dos semiperiodos depende de la resistencia del sensor. Esta señal de salida, Xout(t), sirve tanto para el oscilador de señal como para la entrada a la interfaz de modulación térmica.

La interfaz de modulación térmica está compuesta por un contador digital, seguido de un traductor de nivel de tensión, y finalmente, un amplificador que proporciona la señal de potencia necesaria para accionar la temperatura del sensor.

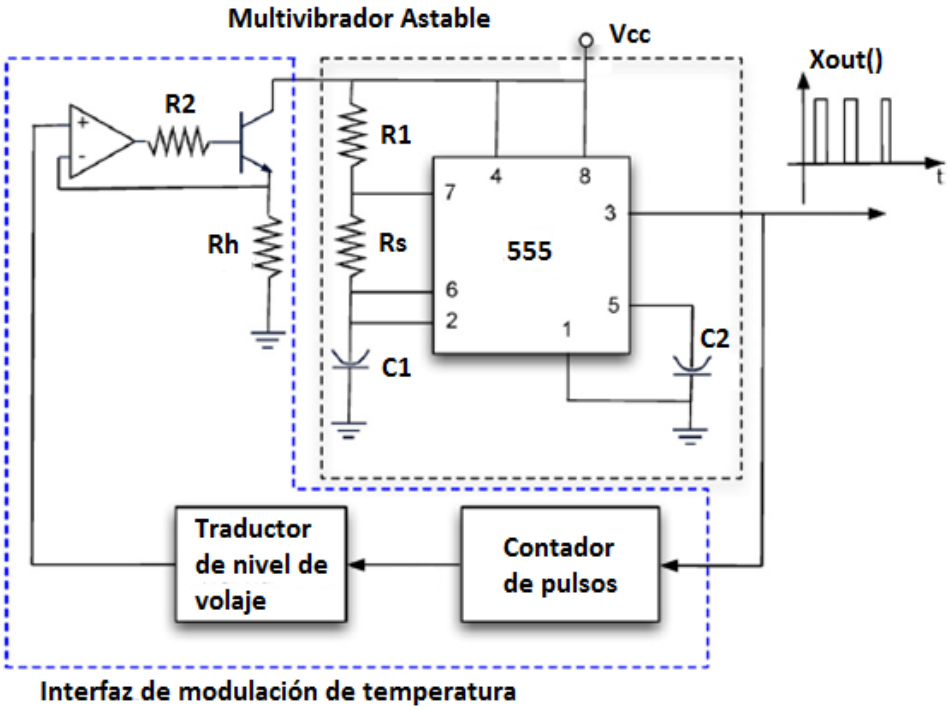


Figura 3. 20: *Esquema circuito self-adapted temperarture modulation. Imagen adaptada de [8].*

Después de un cierto número de pulsos de entrada, se produce una transición entre los dos estados lógicos de la salida del contador digital *Xc(t)*. Para cada uno de los estados lógicos se aplica una tensión al calentador del sensor que se mantendrá constante hasta que se produzca otra transición de *Xc(t)*. De esta forma, el contador digital permite adaptar la modulación en frecuencia de la señal con el fin de satisfacer las propiedades dinámicas del sensor. Es importante tener en cuenta que una transición muy rápida de *Xc(t)* ocasiona temperatura de trabajo sea casi constante y proporcional al valor de la señal *Xout(t)*. Por esta razón la longitud del contador debe ser optimizada. En el caso del trabajo realizado por Martinelli la longitud se fija en $k = 16$, de forma que el oscilador cambiará su señal de salida cada 8 pulsos de entrada. Se aplican tensiones correspondientes a los estados 0 y 1. De esta manera, la modulación de la temperatura se caracteriza por dos semiperiodos que están relacionados con las dos tensiones de calentamiento y el patrón de salida estará compuesto por 16 pulsos.

El circuito modulador de temperatura también establece los valores límite de la tensión de calefacción aplicada durante los ciclos de temperatura. En correspondencia a estas tensiones, después de un tiempo suficientemente largo, la temperatura del sensor alcanzará la temperatura límite (*Tmin*, *Tmax*). Durante el tiempo de medición, el cambio dinámico en la resistencia del sensor se refleja en un patrón de temperatura inestable; cuando se alcanza el estado estacionario, el patrón de pulsos será constante (véase la Figura 3. 21). La longitud del patón de salida *Xout(t)* será función tanto de la frecuencia de la señal de salida como de la longitud del contador. En particular, a mayor longitud de contador, más largos son los pulsos de salida.

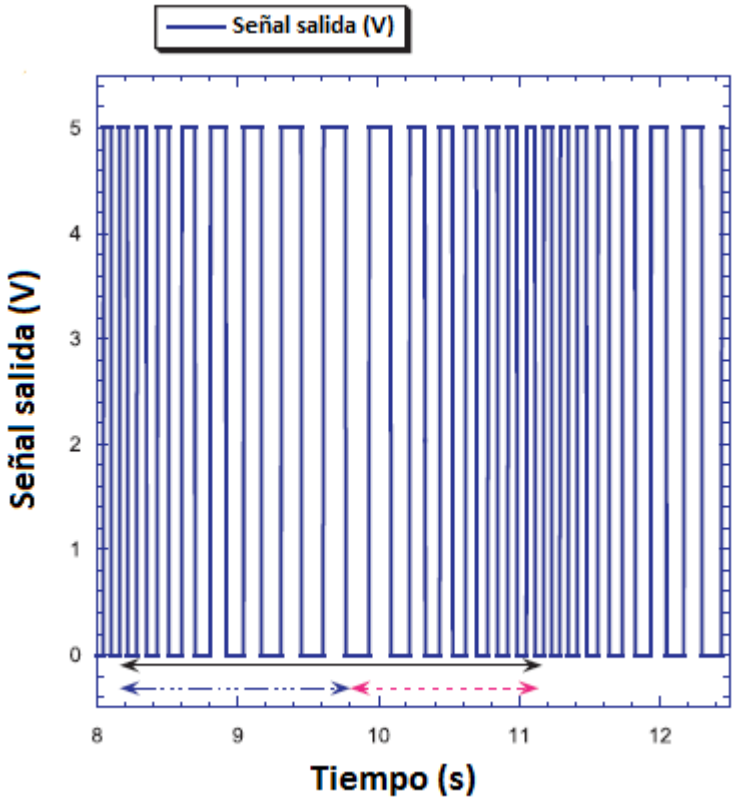


Figura 3. 21: *Señal Xout(t) como patrón de pulsos. Imagen adaptada de [8].*

A la vista de la Figura 3. 22, cuando se produce un cambio de mínimo a máximo en la señal de heater, la temperatura pasa a aumentar hacia *Tmax*, siguiendo el comportamiento exponencial usual que determina la constante térmica del sensor. Antes de que la temperatura alcance su máximo se produce un nuevo cambio, de estado alto a bajo, en la señal de heater que ocasiona que la señal de temperatura disminuya hasta *Tmin*.

Comparando las Figura 3. 21 y Figura 3. 22, se pone en evidencia que el número de impulsos que componen el patrón está igual dividido por los dos semiperiodos de la modulación térmica (8+8), aunque se caracterizan por distintas longitudes de tiempo.

En comparación con la modulación térmica estándar de la temperatura que afecta directamente a la resistencia del sensor en función del gas al que este expuesto. En el método desarrollado por Martinelli, se intercambian los términos, es decir, aquí la resistencia del sensor determina la temperatura de una manera que depende del gas detectado. En el equilibrio, las temperaturas límite y el patrón de pulsos están ambos correlacionados con la calidad y cantidad de odorante. Entonces, si la relación entre la temperatura y a resistencia depende del gas al que está expuesto el sensor, debe ser posible seleccionar el rango óptimo de temperaturas y el tamaño del contador digital, para obtener diferentes patrones de pulsos que puedan discriminar odorantes.

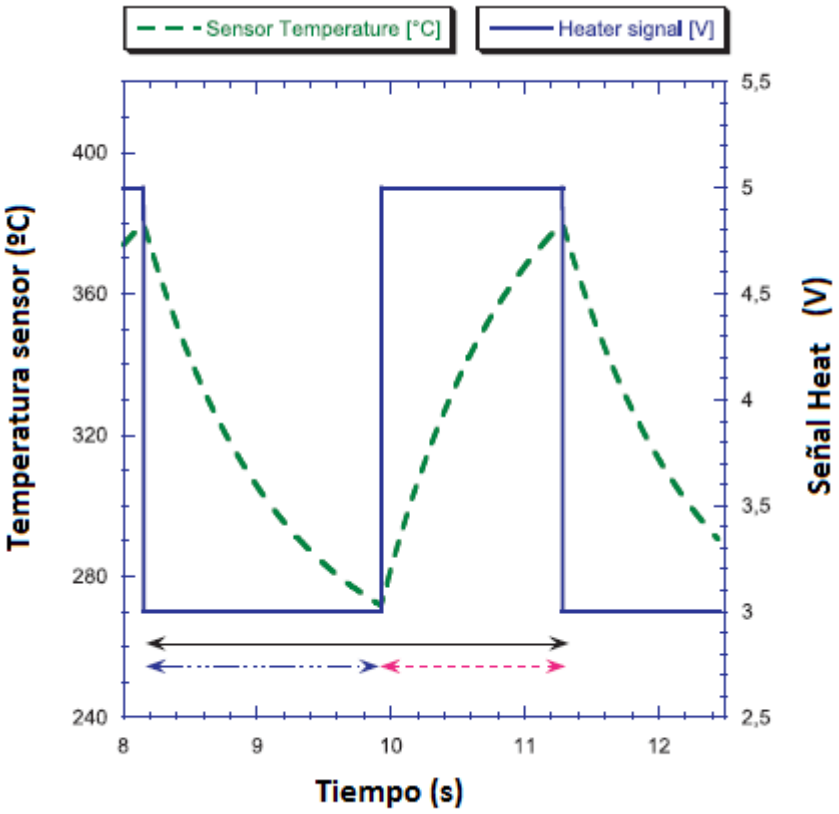


Figura 3. 22: *Representación de la señal Xout(t) y la temperatura del sensor (ºC). Imagen adaptada de [8].*

3.5.1. Circuito multivibrador. NE555

Para el circuito multivibrador utilizado en la técnica de auto-modulación de temperatura se ha elegido el componente NE555, con datasheet [17]**¡Error! No se encuentra el origen de la referencia.** Se trata de un circuito integrado que funciona como multivibrador y puede ser configurado en dos modos, monoestable y astable. En el anexo [C], se puede encontrar más información sobre teoría de multivibradores y la configuración de estos.

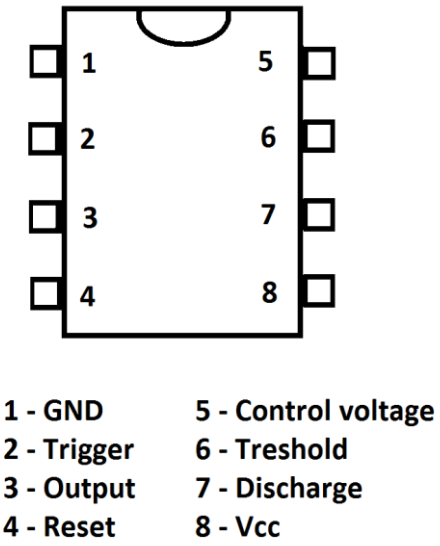


Figura 3. 23: *Circuito integrado 555. Pines de conexión.*

Como se puede observar en el datasheet del componente [17] , el integrado tiene ocho pines de conexión:

- **GND**. Es el polo negativo de la alimentación, generalmente tierra (masa).

- **Trigger (Disparo)**. Es donde se establece el inicio del tiempo de retardo si el 555 es configurado como monoestable. Este proceso de disparo ocurre cuando esta patilla tiene menos de 1/3 del voltaje de alimentación. Este pulso debe ser de corta duración, pues si se mantiene bajo por mucho tiempo la salida se quedará en alto hasta que la entrada de disparo pase a alto otra vez.

- **Output (Salida)**. Salida que muestra el resultado de la operación del temporizador, que sea que esté conectado como monoestable, estable u otro. Cuando la salida es alta, el voltaje será el voltaje de alimentación (Vcc) menos 1.7 V. Esta salida se puede poner a cero voltios con la ayuda de la patilla de reinicio.

- **Reset (Reinicio)**. Si se pone a un nivel por debajo de 0.7 voltios, la patilla de salida se activa a nivel bajo. Si por algún motivo esta patilla no se utiliza hay que conectarla a alimentación para evitar que el temporizador se reinicie.

- **Control de voltaje**. Cuando el 555 se utiliza en modo de controlador de voltaje, el voltaje en esta patilla puede variar casi desde Vcc (en la práctica Vcc – 1.7 V) hasta casi 0 V (aprox. 2 V menos). Así es posible modificar los tiempos.

- **Threshold (Umbral)**. Es una entrada a un comparador interno que se utiliza para poner la salida a nivel bajo.

- **Discharge (Descarga)**. Utilizado para descargar con efectividad el condensador externo utilizado por el temporizador para su funcionamiento.

- **Voltaje alimentación (Vcc)**. Es la patilla donde se conecta el voltaje de alimentación que va de 4.5 V hasta 16 V.

3.6. Circuito de adquisición de odorante

El circuito de adquisición de odorante consta de varias partes que se pueden ver en la siguiente figura y que se detallan más adelante. Básicamente, será necesario un motor de succión y un circuito de electroválvulas que conmuten el paso del odorante deseado desde la matriz de muestras. Es importante la ubicación de cada componente dentro del circuito, esto se detallará en la sección [4.5] de la memoria.

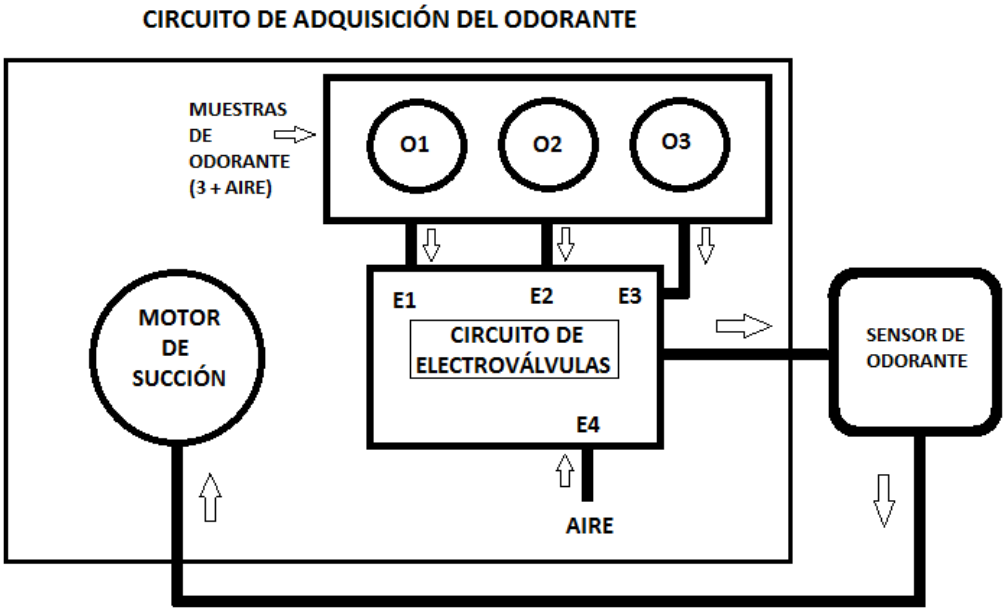


Figura 3. 24: *Esquema de circuito de adquisición de odorante*

3.6.1. Motor de succión

Dentro del sistema de adquisición, el elemento más importante es el motor de succión. La elección del mismo no puede ser aleatoria, se necesita un elemento robusto, pero pequeño y con una durabilidad prolongada, pues para algunas experimentaciones podría estar funcionando durante días sin parar. Además, es conveniente que su consumo sea mínimo.

Siguiendo estas premisas, se ha optado por un motor, o, mejor dicho, microbomba de succión de la casa RS, en particular, el D250s Micropump Range [22].

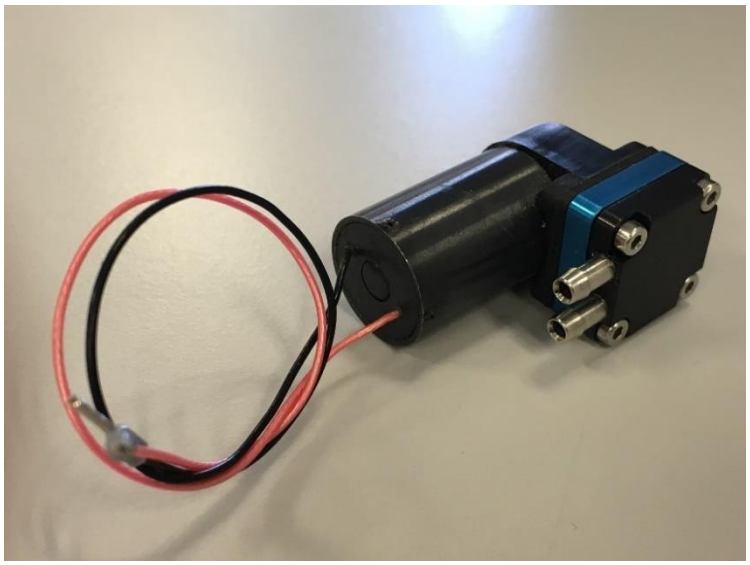


Figura 3. 25: *Microbomba de succión D250s de RS.*

Las microbombas de succión de RS son bombas de cebado automático en miniatura de alta calidad para líquidos y gases. Son muy eficientes, pequeñas y ligeras. La construcción sólida y su alta tolerancia a la temperatura, permiten usarlas de forma fiable incluso en condiciones adversas.

La microbomba puede ser alimentada con baterías, con una fuente DC de hasta 4.5V y con una fuente de alimentación ajustable. La velocidad de bombeo puede ser ajustada mediante variación del voltaje de entrada entre 3.0V y 4.5V.

En el siguiente cuadro se muestran las principales características del motor de succión:

Medio	Voltaje entrada (V)	Consumo (W)	Presión máx. (psi)	Flujo máx. (mlpm)
Aire/Gas	3.0 – 4.5	0.1 – 0.6	11	580
Agua	3.0 – 4.5	0.6 – 1.8	9	90

Tabla 5: *Tabla de características Microbomba 250s RS.*

El control del motor se realizará a través de un circuito de adaptación en potencia que se explica en la sección [5.2.5] y su código de control asociado en la sección [5.3.1.2]. Se realizará de tal forma que se pueda controlar la potencia de succión del motor, hecho que podría derivar en futuros experimentos controlando el flujo de aire del circuito, modulación por flujo.

3.6.2. Circuito de electroválvulas

El circuito de electroválvulas a diseñar constará de 4 electroválvulas idénticas. Se utilizarán las mismas que uso David Yañez en su publicación [2]. Se trata de la electroválvula SY114-VLOZ-Q de la empresa SMC [18]. Es una electroválvula de solenoide de 3 vías, 2 posiciones y normalmente cerrada (NC). En la siguiente figura se muestra la electroválvula y su diagrama de flujo.

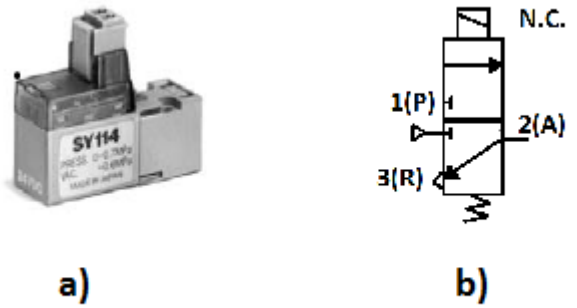


Figura 3. 26: *a) Electroválvula de solenoide SY114-VLOZ-Q. b) Diagrama de flujo electroválvula 3/2 NC*

Cuando la electroválvula se encuentra sin energía, no se le aplica corriente, se encuentra en posición cerrada. En este caso permite el paso de líquido o gas entre el orificio 2(A) y 3(R). Cuando se les aplica energía, la electroválvula se conmuta y permite el paso entre los orificios 1(P) y 2(A).

Las características principales de este componente son:

- Alimentación a 6Vdc ± 10%
- 3 vías.
- 2 posiciones.
- Normalmente Cerrado.
- Presión de funcionamiento entre 0 y 0.7 MPa.
- Temperatura de funcionamiento entre -10ºC y 50ºC.

Para la elección de este componente fue preciso estudiar los fundamentos de electroválvulas (véase el Anexo [D]), para entender el funcionamiento de estas, y una comparación exhaustiva de las diferentes electroválvulas del mercado. En la sección [4.5] se expondrá el diseño realizado para el circuito de electroválvulas y se detallará la lista de componentes necesarios para realizarlo.

Será necesario también crear un circuito de adaptación en potencia para el control de las electroválvulas que se detalla en la sección [5.2.4] y el software de control que se explica en la sección [5.3.1.1].

3.7. Circuito adquisición temperatura y humedad

Como ya se indicó en el apartado [3.3.1.2] la temperatura y la humedad ambiente influyen directamente en la sensibilidad del sensor de odorante. Por esto, y dado que es muy costoso realizar las pruebas en un espacio controlado, es conveniente realizar una adquisición de valores de temperatura y humedad durante el transcurso de cada experimentación.

En trabajos anteriores del GNB como en [4], el sensor de temperatura y humedad utilizado ha sido el DHT11. Para el presente proyecto se descarta el uso de este componente, puesto que ya se comercializa una nueva y mejorada versión del mismo.

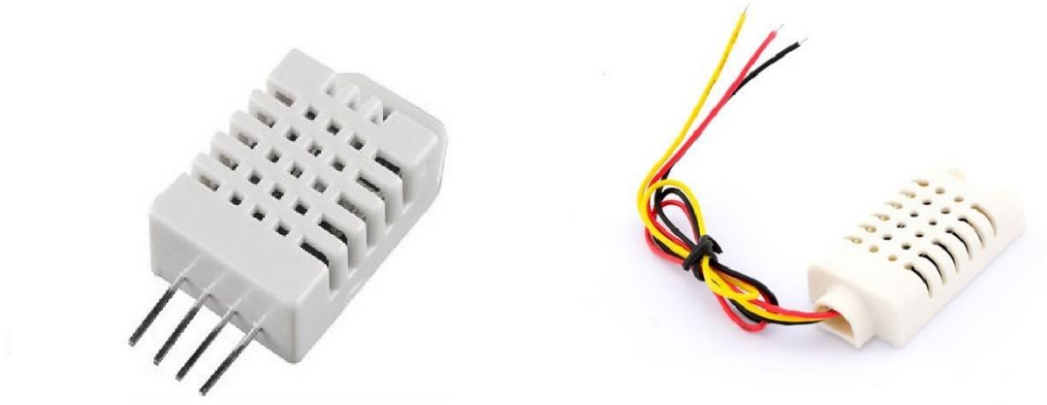


Figura 3. 27: *Sensor de temperatura y humedad DHT22/AM2302 Standad (izquierda) y DHT22/AM2302 Wiring (derecha).*

El DHT22 o AM2302, se comercializa en distintos encapsulados, nos centraremos en dos. El primero, denominado Standard, viene suelto, con su funda blanca plástica y cuatro pines de conexión (a la vista en la imagen izquierda de la Figura 3. 27). El segundo, denominado Wiring, es la versión cableada, el sensor esta soldado a una placa, contiene una

resistencia pull-up de entre 3-6KΩ, un condensador de filtrado de 100nF y está cubierto por una funda blanca de mayores dimensiones (a la vista en la imagen derecha de la Figura 3. 27). Son la versión actualizada con mayores rangos de temperatura y humedad, y más precisión.

3.7.1. Prestaciones

Sus prestaciones generales son:

- Alimentación: 3.3v -5-5v, tomando como valor recomendado %v.
- Resolución decimal, es decir, los valores tanto para la humedad como para la temperatura serán números de una cifra decimal.
- Tiempo de muestro: 2 segundos. Sólo pude ofrecernos datos cada 2 segundos.

En cuanto a adquisición de temperatura, sus características son:

- Rango de valores de -40ºC hasta 80ºC de temperatura.
- Precisión: ±0.5ºC, ±1ºC como máximo en condiciones adversas.
- Tiempo de respuesta: <10 segundos, es decir, de media tarda menos de 10 segundos en reflejar un cambio de temperatura en el entorno.

En cuanto a adquisición de humedad se refiere, sus características son:

- Rango de valores desde 0% hasta 99.9% de humedad relativa.
- Precisión: ±2%RH, a una temperatura de 25ºC.
- Tiempo de respuesta: <5 segundos, es decir, de media tarda menos 5 segundos en reflejar una variación de la humedad relativa en el entorno.

3.7.2. Conexión

En función de su tipo de encapsulado el sensor de temperatura y humedad se conecta de distinta forma. El Standard, que tiene cuatro pines, tiene el pin 1 de alimentación, pin 2 es la salida del sensor, el pin 3 es libre y el pin 4 es GND. El Wiring, sin embargo, tiene el cable rojo es alimentación, el cable negro a GND y el amarillo es la salida del sensor.

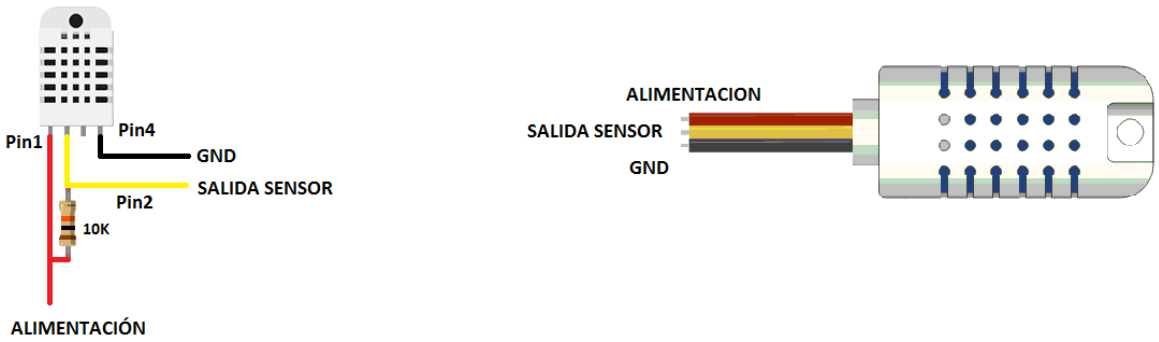


Figura 3. 28: *Conexionado del sensor de temperatura y humedad DHT22 Standar (izquierda) y Wiring (derecha).*

En la figura se observan las diferentes conexiones que tiene cada encapsulado. Estas diferencias se deben a que el Standard para funcionar correctamente necesita de una resistencia pull-up a su salida. En el Wiring, sin embargo, no es necesario hacer esto puesto que la resistencia ya viene integrada en el encapsulado, junto con un condensador de filtrado. De forma opcional, con el encapsulado Standard, podemos también filtrar la alimentación colocando un condensador de 100nF entre la alimentación y GND.

Más información del componente se puede ver en su datasheet [23]. Para su utilización se ha elaborado el código de la sección [5.3.1.3].

4. Metodología y tecnologías utilizadas

4.1. Introducción

En esta sección se detallan los pasos seguidos para la creación de la plataforma de experimentación, desde el diseño de la estructura de montaje hasta la configuración de la placa BBB. Pasando por el arranque de la BBB por primera vez, una breve introducción al lenguaje de programación que se va a utilizar o las librerías necesarias.

Este apartado servirá, además, como guía para futuros alumnos del GNB que realicen futuros proyectos o investigaciones con la presente plataforma de experimentación. Ya que, como se indicó, el objetivo de este proyecto es la consecución de una plataforma versátil y de fácil integración de algoritmos para la discriminación de odorantes. Será de gran ayuda lo expuesto en los siguientes apartados para entender el funcionamiento de la plataforma y poder evolucionarla en trabajos futuros.

Adquiere vital importancia está sección en caso de algún problema con alguno de los elementos de la plataforma. Imaginemos que la placa BBB se quema y hay que sustituirla, a la persona que le suceda esto le será de gran ayuda lo expuesto a continuación. Además, es conveniente leer el libro [24] *Getting started with BeagleBone*, donde se detallan con mayor exactitud los expuesto en esta sección.

4.2. Toma de contacto con la BBB

Tal y como se indicó en la sección [3.2], el sistema embebido elegido para la plataforma es la placa BeagleBone Black. Es fácil de adquirir a través de la empresa Element14 (la fuente de alimentación de 5V a 1ª debe comprarse por separado).

La tarjeta viene precargada con un sistema operativo, dependiendo de la versión puede venir con Linux Debian o Linux Angstrom. Con bastante seguridad, la versión precargada no será la más actual, por lo que lo mejor es actualizar.

Antes de realizar el primer arranque conviene saber que hay dos formas de utilizar la tarjeta, esta viene con conector de monitor y USB, al que si se le incluye un HUB permite conectar un ratón y un teclado; y con ello utilizar la tarjeta como si de un ordenador se tratase. O se puede utilizar como esclavo a través de SSH (Secure Shell) mediante el puerto USB mini que tiene la tarjeta.

Figura 4. 1: *BeagleBone Black, cable USB y Transformador DC a 5V.*

En caso de disponer de monitor, teclado y ratón, la tarjeta puede utilizarse directamente. Aunque se recomienda, si es la primera vez que se usa, realizar antes una actualización de sistema operativo (se detallará más adelante como actualizar). Para la realización del proyecto se ha utilizado la tarjeta en modo esclavo.

Pero antes de empezar a usar el sistema embebido conviene conocer cómo se puede destruir la BeagleBone.

4.2.1. Precauciones con BBB

La BeagleBone Black es un complejo y delicado componente que es muy fácil dañar si no se manipula con cuidado. El microprocesador no es fácil de remplazar y si se daña, es necesario comprar una placa nueva.

A continuación, una lista con las cosas que no se deben hacer:

- No apagar la BBB quitando el jack de alimentación o la alimentación USB directamente. La BBB debe ser apagada correctamente, utilizando software de apagado.

- No apoyar la BBB sobre superficies metálicas, superficies conductoras o sobre resistencias. Por esto es aconsejable comprar o fabricas una cape (o caja) para aislar la tarjeta y evitar el contacto de sus conexiones con alguna superficie conductora.
- No conectes circuitos, como entrada o salida de los pines P8/P9, provenientes de otra fuente de corriente sin antes asegurarte de que dicha corriente de entra, salida entra dentro de los límites de la BBB. La corriente máxima de entrada no debe ser de 4-6 mA, mientras que la corriente máxima a la salida será 8 mA.

- Los GPIOS toleran 3.3V y los ADC toleran 1.8V. No se debe conectar directamente a la BBB un circuito que esta alimentado a 5V o se destruirá la placa.

- No conectar a los pines P8/P9 circuitos que apliquen corriente mientras la BBB está apagada.

Igualmente, que hay cosas que se deben evitar hacer al manipularla BBB, también, hay procedimientos que conviene realizar siempre:

- Comprobar siempre cuidadosa mente los números de pin que se utilizan. Hay 46 pines en cada cabecera, y es fácil equivocarse y conectar el pin 30 cuando se trataba de conectar el 32. Por lo que se recomienda siempre contar dos veces los pines antes de conectar.

- Antes de conectar nuevos y complejos circuitos a la BBB leer el SRM (System Reference Manual) [25] y los otros manuales que se indican en referencias [24][26][27]

Además de los manuales indicados, se ha creado un manual de requisitos necesarios para el uso y correcto funcionamiento de la plataforma, se puede observar en el Anexo [E]. En él se encuentran los primeros pasos a realizar en las primeras tomas de contacto con la plataforma, se indican los requisitos de software necesarios, así como su instalación, formas de conexión con la tarjeta y de esta a internet, o los paquetes y librerías necesarias.

4.3. Precauciones con dispositivos de E/S

En la BeagleBone Black, el acceso a los dispositivos de E/S habituales delos sistemas Linux (consola, puerto serie, conexión Ethernet, etc) es igual que en otros equipos con el mismo S.O.

No obstante, los dispositivos de entrada/salida más específicos, agrupados bajo el nombre GPIO (*General Purpose Input Output*), E/S digital, entrada analógica, salida PWM; requieren un trabajo más laborioso, que con lleva la lectura de documentación dispersa [24][25][26][27] y en muchos casos sólo existente a partir de respuestas en foros.

Hay dos conjuntos de pines de expansión, P8 y P9, en cada uno de estos pines se exponen múltiples señales de la CPU, por lo que existe un multiplexor que permite seleccionar cuál

de las señales es accesible en cada momento. En el Anexo [B] se encuentran las tablas con los distintos modos de funcionamiento de cada pin.

A continuación, se reúnen los requerimientos y procedimientos para estas E/S.

4.3.1. E/S digitales

Es posible utilizar múltiples pines para E/S digital a través de los conectores P8 y P9. Estos pines están indicados con GPIOx[xx] en el modo 7 de las tablas del Anexo [B].

¡Importante! Estos pines funcionan a un voltaje de 3.3 V, hay que tener cuidado porque se dañan con un valor de tensión mayor. Cuando se configuran como salida no se les puede solicitar una corriente superior a los 6 mA, o también se dañará la tarjeta.

La configuración de un pin para E/S digital permite seleccionar si se desea entrada o salida, y si se activa una resistencia interna pull-up o pull-down. La configuración se realiza sobre los bits 4-7 del modo, según la siguiente tabla:

BIT	7	6	5	4	3	2	1	0
Valor	0	0	1=in 0=out	0=pulldown 1=pullup	0=pullenable 1=pulldisable	1	1	1

Tabla 6: *Bits de configuración E/S digital de BBB.*

Para la activación y desactivación de estos pines se ha instalado la librería Adafruit_BBIO [E.8.1.1], que contienen funciones para el control y setup de los pines.

- GPIO.setup (pin, mode, pull_up_down)

Donde pin es el GPIO utilizado escrito en formato P8_xx o P9_xx, mode es GPIO.OUT o GPIO.IN (salida y entrada, respectivamente). Pull_up_down puede tomar tres valores; GPIO.PUD_OFF, GPIO.PUD_DOWN o GPIO.PUD_UP, resistencia apagada, valor bajo o valor alto, respectivamente.

Por ejemplo, para configurar el pin P8_12 como salida, con valor y alto y resistencia de pull up, la instrucción sería:

- GPIO.setup ('P8_12', GPIO.HIGH, GPIO.PUD_UP)

La biblioteca GPIO tiene un método bloqueante, denominado GPIO.wait_for_edge, que permiten detectar transiciones entre pulsos. Esto quiere decir que si el valor es de caída (paso de 3.3V a 0V), de subida (cambia de 0V a 3.3V) o las dos opciones (que detecte subidas o bajadas), la biblioteca GPIO lo detectará y continuará la ejecución del programa.

- GPIO.wait_for_edge("P8_14", GPIO.RISING) ➡ Espera pulso de subida.
- GPIO.wait_for_edge("P8_14", GPIO.FALLING) ➡ Espera pulso de bajada.

- GPIO.wait_for_edge("P8_14", GPIO.BOTH) ➡ Espera subida o bajada.

Otra opción que ofrece y no es bloqueante es añadir un evento a detectar, GPIO.add_event_detect. Primero hay que configurar el evento a detectar, entonces ya se puede hacer cualquier otra cosa que el programa deba hacer, y más adelante, se comprueba si el evento ha sido detectado.

4.3.2. Entradas analógicas

La BeagleBone Black cuenta con un total de 7 entradas analógicas en los pines denominados con AINxx en el conector P9. Cuentan con una resolución de 12 bits, es decir, puede leer de 0 a 4095 valores. Teniendo su tiempo de muestreo en 125ns.

Un valor de tensión de entrada de 1.8 V el puerto ADC lo lee como 4095. Si por ejemplo el valor de lectura es 2567, la tensión de entrada se calcula según la siguiente fórmula:

$$\frac{Valor\ Lectura}{4096} \times 1.8V \qquad \frac{2567}{4096} \times 1.8V = 1.28V$$

¡Importante! Estos pines funcionan a un voltaje de entre 0 y 1.8 V, aplicando un voltaje fuera de ese rango la placa se dañará. Además, la corriente máxima de entrada no debe superar los 2 μA.

Para la utilización de estos pines se ha instalado la librería Adafruit_BBIO [E.8.1.1], que contienen funciones para el control y setup de los pines. La librería ofrece tres métodos: uno de configuración y dos de lectura.

- Configuración: antes de realizar una lectura por puerto ADC es necesario realizar una configuración previa o si no se producirá un error en la lectura.

```
ADC.setup()
```

- Lectura: devuelve el valor entre 0 -1.0, es decir, con este método para saber el voltaje leído es necesario multiplicar el valor obtenido por 1.8V

```
value = ADC.read("P9_40")
voltaje = value * 1.8V
```

- Lectura de voltaje: devuelve el valor no normalizado, el valor de salida corresponde al voltaje en mV.

```
value = ADC.read_raw("P9_40")
```

Antes de cualquier lectura es necesario ejecutar el setup de los puertos ADC. Además, hay que tener en cuenta que hay un pequeño error en el controlador ADC por lo que es necesario leer los valores en dos ocasiones con el fin de obtener el valor más reciente.

4.3.3. Salidas PWM

La tarjeta BeagleBone Black presenta 8 salidas PWM, estas funcionan como se indicó en la sección [3.2.2]. Los pines PWM son los siguientes:

- EHRPWM0A: pin P9_22 (modo 3) o pin P9_31 (modo 1).
- EHRPWM0B: pin P9_21 (modo 3) o pin P9_29 (modo 1).
- EHRPWM1A: pin P8_36 (modo 2) o pin P9_14 (modo 6).
- EHRPWM1B: pin P8_34 (modo 2) o pin P9_16 (modo 6).
- EHRPWM2A: pin P8_19 (modo 4) o pin P8_45 (modo 3).
- EHRPWM2B: pin P8_13 (modo 4) o pin P8_46 (modo 3).
- ECAPPWM0: pin P9_42 (modo 0).
- ECAPPWM2: pin P9_28 (modo 4).

Para que estas salidas funcionen es necesario activar previamente el reloj de las mismas, escribiendo adecuadamente en las posiciones del mapa de E/S. Esto se realiza por medio de la librería disponible [E.8.1.1].

¡Importante! Estos pines funcionan a un voltaje de 3.3 V, hay que tener cuidado porque se dañan con un valor de tensión mayor. Cuando se configuran como salida no se les puede solicitar una corriente superior a los 6 mA, o también se dañará la tarjeta.

Para la utilización de estos pines se utilizan las funciones suministradas por la librería Adafruit_BBIO:

- PWM.start (channel, duty, freq=2000, polarity=0)

  El método start activa la PWM del pin introducido en channel con un ciclo de trabajo dado por duty, este último valor debe estar entre 0.0 y 100.0. Este método, además, aunque también funciona sin ellos, permite cambiar la frecuencia y la polaridad que vienen por defecto.

  Tanto la frecuencia como el ciclo de trabajo se puede variar en cualquier momento usando los siguientes métodos.

- PWM.set_duty_cycle (channel, duty)
- PWM.set_frequency (channel, frequency)

  Una vez se han terminado de usar los pines PWM conviene deshabilitar el canal y limpiarlo, esto se hace con los dos siguientes métodos:

- PWM.stop(channel)
- PWM.cleanup()

4.3.4. Resumen de todas las precauciones y protecciones consideradas para proteger BBB

Durante el desarrollo de la plataforma han ido surgiendo problemas para poder medir las señales con la tarjeta BBB. Como se ha venido indicando hay ciertas limitaciones en cuanto a corrientes y voltajes de salida o entrada a la BeagleBone. Durante el desarrollo de estos circuitos se quemó una tarjeta y tuvimos que comprar otra y mejorar los sistemas de protección. A continuación, se va a exponer un listado de las protecciones utilizadas y las secciones en las que se encuentran explicadas.

- Resistencia en serie de 1 kΩ en todos los pines, tanto de entrada como de salida utilizados en la BBB. Véanse las secciones [3.4], [3.5], [3.6.1] y [3.6.2]. Estas resistencias evitan que se produzcan corrientes de entrada a la BBB.

- Resistencias de pull-up y pull-down internas de la BBB. Como se ha indicado en 4.3.1 los puertos GPIO cuentan con una resistencia cuyo valor se puede configurar por software y que evita corrientes no deseadas de entrada.

- ULN2003. Se han utilizado también este componente como mecanismo de protección, en las secciones [3.5] y [3.6.2] se ha explicado su funcionamiento. El componente contiene un array de transistores Darlington que se alimentan a la corriente deseada (3.3 V, por ejemplo, para no quemar la placa) y que al recibir una señal de un voltaje superior al soportado por la BBB lo transforma al voltaje de alimentación.

- Divisor de tensión. Se han utilizado en los circuitos de las secciones [5.2.1] y [5.2.3], es una forma fácil y segura de reducir los voltajes que se desean medir.

4.4. Diseño y montaje plataforma

En la presente sección se detalla el diseño y montaje de la estructura de la plataforma. Para la elaboración de la misma se ha utilizado madera de tipo DM de 3 mm de espesor, elementos de tornillería, alambre de sujeción y piezas impresas.

En la siguiente figura se presenta el plano con las dimensiones de la estructura. Esta está compuesta por 3 módulos.

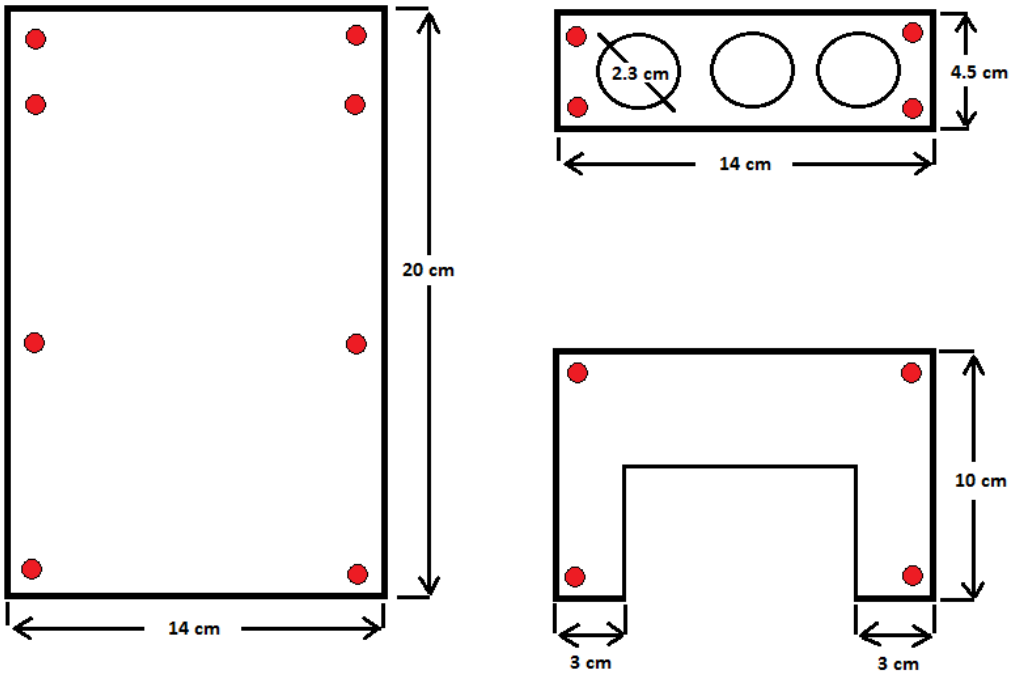


Figura 4. 2: *Plano de la estructura de la plataforma de adquisición de odorante.*

Donde se diferencian tres partes, la pieza izquierda es la base de la plataforma. Es una pieza de madera DM de 20 centímetros de largo por 14 centímetros de ancho. En ella se fijarán la BBB, las electroválvulas, el motor de succión y el sensor. Las dos piezas de la derecha se colocarán una altura por encima de la base. La pieza superior tiene 3 agujeros de 2.3 centímetros de diámetro que se utilizarán como sujeción para las probetas con muestras de odorante. Por último, la pieza inferior derecha servirá para fijar una protoboard para realizar circuitos electrónicos necesarios.

En la Figura 4. 2 se destacan en rojo los agujeros de unión de las piezas, estos se realizaron con una máquina Dremel y tiene 3 mm de diámetros. Para la unión se utilizan 8 tornillos de 4 cm de longitud, 16 arandelas y 24 tuercas.

La Figura 4. 3 muestra las tres piezas que forman la estructura principal de la plataforma, se pueden observar los tornillos de unión, ya fijados en la pieza base. Además, se observan unos pequeños agujeros (de aproximadamente 1 mm de diámetro) marcados en amarillo y otros dos (de aproximadamente 2 mm de diámetro) marcados en verde. Los primeros servirán para fijar con alambre el circuito de electroválvulas y los segundos para fijar la BBB.

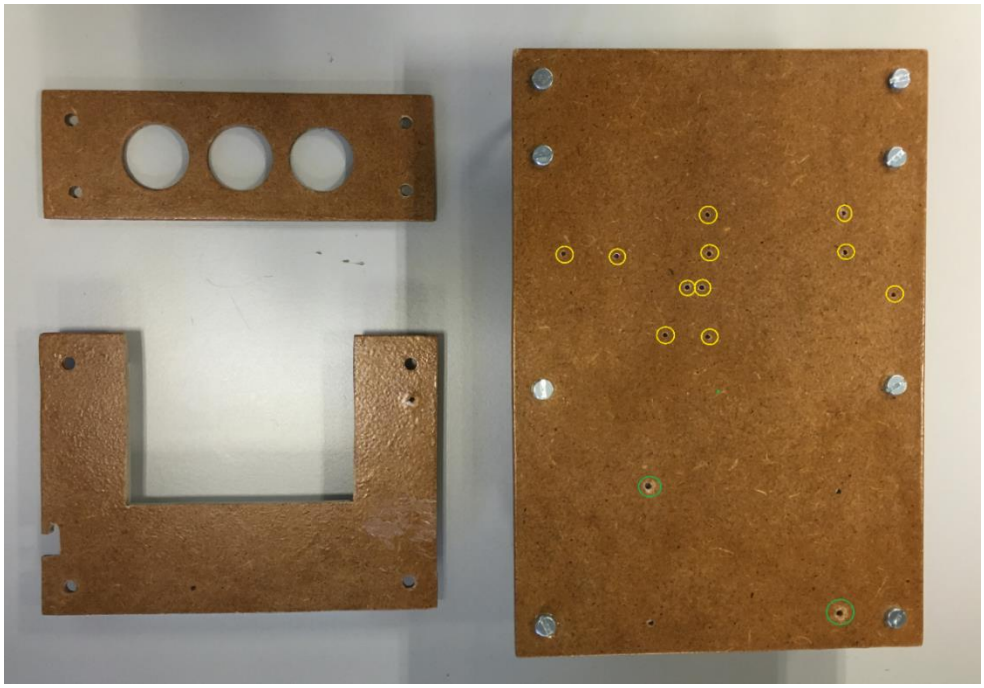


Figura 4. 3: *Piezas de DM de la estructura de la plataforma.*

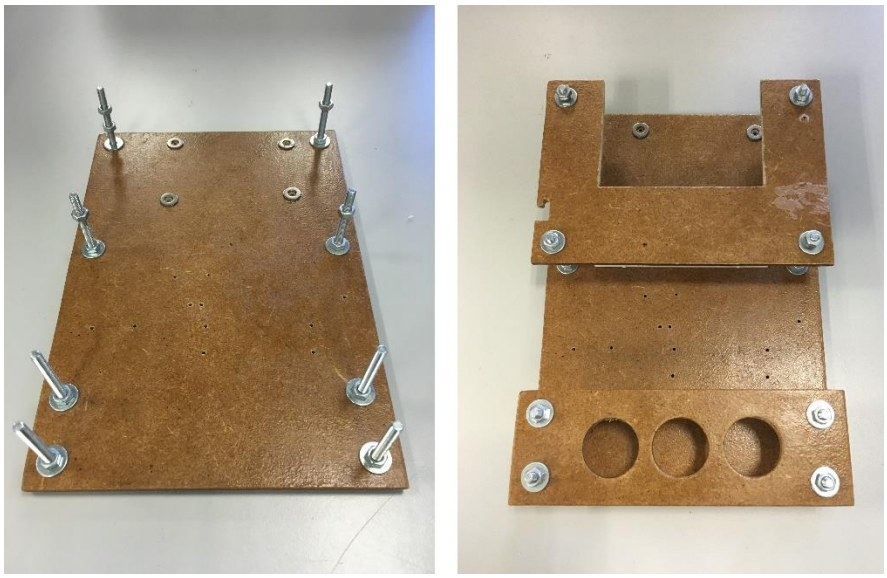


Figura 4. 4: *Base de la estructura con sus tornillos de montaje (a la izquierda de la figura). Estructura de madera montada (a la derecha de la figura).*

La plataforma se ha montado dejando un espacio de unos 2.5 centímetros entre la base y las maderas superiores con el fin de facilitar el acceso a los componentes que se emplacen en la base y para que las probetas con odorante se asienten bien los agujeros realizados para las mismas.

Para no fijar la BBB directamente a la base de madera se ha impreso con una impresora 3D una caja para la tarjeta. El diseño de la caja se ha conseguido a través de la web

http://www.thingiverse.com, donde se pueden encontrar gran variedad de diseños de impresión libres. En la siguiente imagen se muestra la caja ya impresa, su localización y fijación en la estructura.

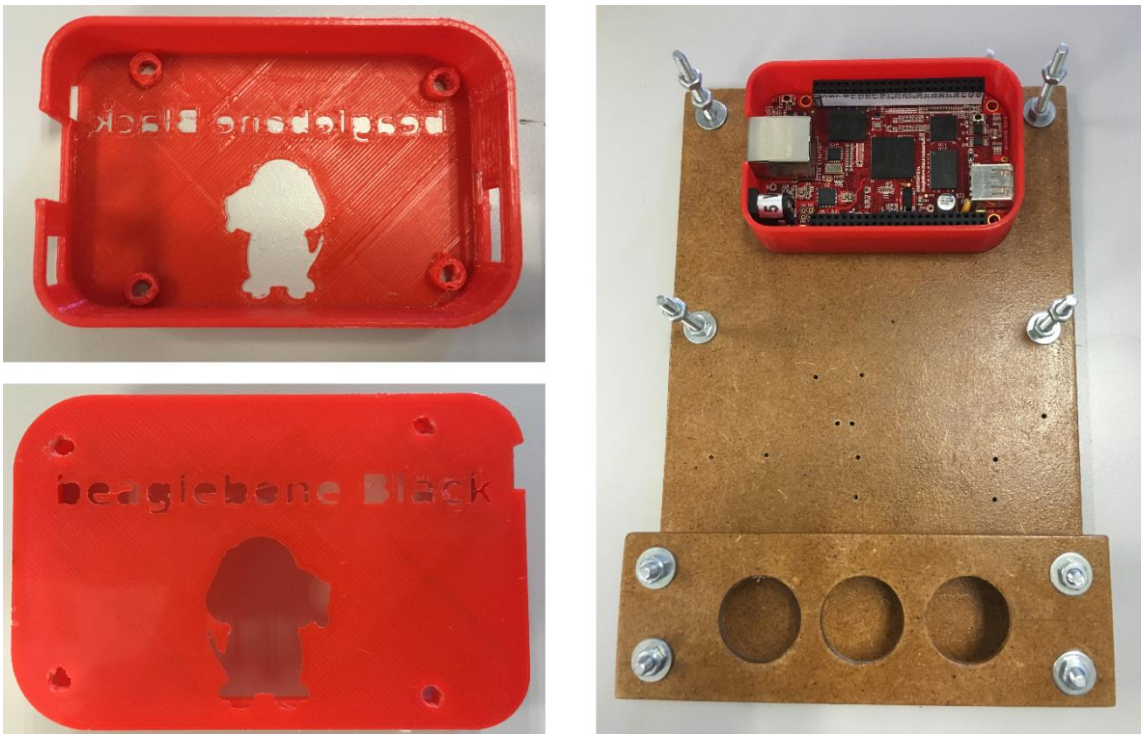


Figura 4. 5: *Caja impresa para BBB. A la izquierda se muestra la pieza impresa por el frente y reverso. A la derecha la BBB fijada en la estructura de madera.*

Por último, se ha incorporado a la estructura una protoboard de tamaño reducido, con 400 puntos de conexión. La protoboard elegida en la parte de atrás tiene un autoadhesivo, un largo de 8.2 cm y ancho de 5.2 cm. Permite ser plegada y separarla en tres partes. En la Figura 4. 6 se puede observar la protoboard, ensamblada, desensamblada y montada en la estructura de madera.

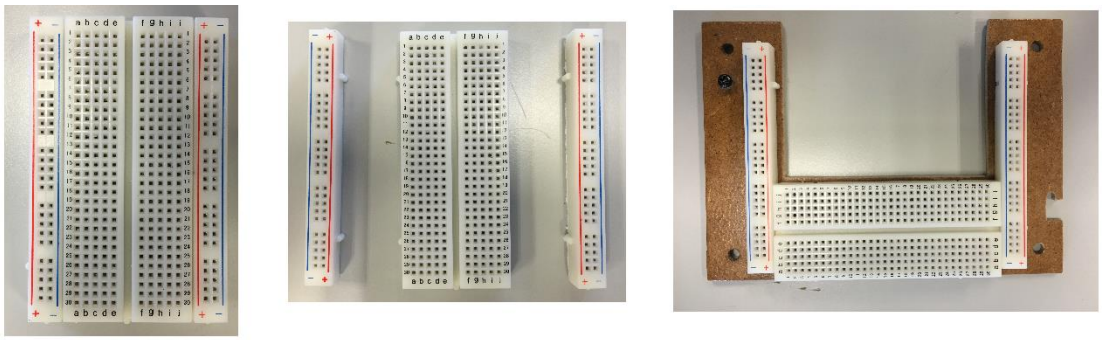


Figura 4. 6: *Protoboard de 400 puntos. Montaje en la estructura de madera.*

4.5. Conexión del circuito de adquisición de odorante

El circuito de adquisición de odorante, tal y como se indicó en la Figura 3. 24, está compuesto por el motor de succión, un circuito de electroválvulas, un muestrario de odorantes y el sensor de odorante. Para la interconexión de los elementos se han utilizado tubos de plástico inerte de dos tipos, de 3 mm y 6 mm de diámetro.

Como el motor se va a utilizar en su posición de succión, este se tiene que ubicar al final del circuito de adquisición. Desde su salida irá un tubo de 6 mm hasta el sensor (que se encuentra encapsulado en una caja hermética diseñada a su medida, ver figura). El sensor se conectará a las electroválvulas y cada una de las electroválvulas saldrá un tubo de 3 mm que conectará con cada tubo de odorante.

A continuación, se detalla una lista de los componentes utilizados para la realización del circuito de adquisición de odorante, que se pueden observar en la Figura 4. 7.

Figura 4. 7: *Componentes del circuito de adquisición de odorante.*

- 4 Electroválvulas SY114-VLOZ-Q [18].
- 4 Bases de montaje de electroválvula.

- Motor de succión RS D200 [22].
- Tubo de plástico inerte de 6 mm de diámetro.
- Tubo de plástico inerte de 3 mm de diámetro.
- Caja de plástico inerte y hermética para el sensor.
- 3 Agujas.
- 3 Agujas con acople para conexión de tubos de 3 mm.
- 7 Conexiones para tubo de 3 mm.
- 3 Tubos de ensayo con tapón de cierre.

4.5.1. Conexión entre las electroválvulas

Como se indicó en la sección [3.6.2] las electroválvulas elegidas tienen 3 vías y dos posiciones. Esto quiere decir, que tienen tres entradas o salidas, pero sólo dos modos de conmutación entre ellas. Las entradas/salidas se denominan A, R y P. En la Figura 4. 8 se muestra la válvula utilizada y su base de montaje.

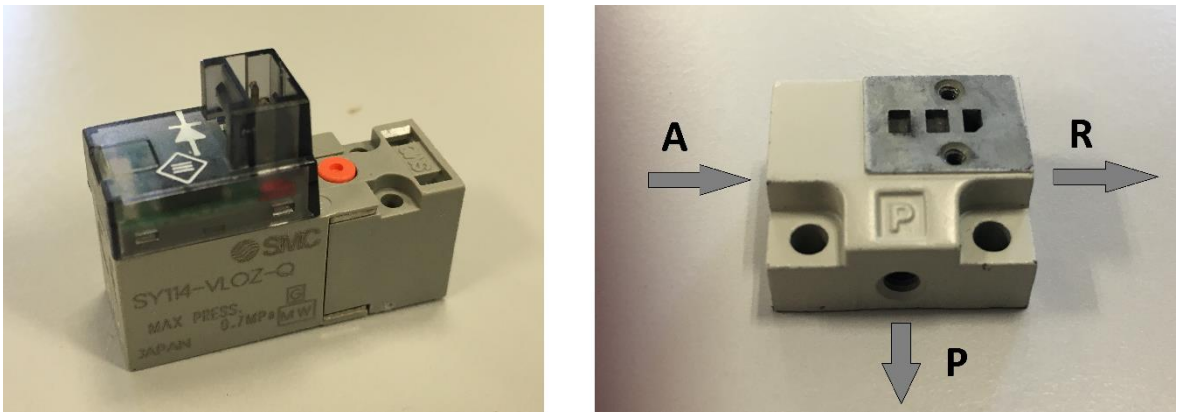


Figura 4. 8: *Electroválvula SY114-VLOZ-Q y base de montaje. Vías de conexión.*

La válvula puede encontrarse en dos estados, estado con energía o sin energía. El primero permite el paso entre P y R, mientras que el segundo permite el paso entre A y R. La conmutación entre A y P no es posible. En la Figura 4. 9 se muestra como conmuta la base de la electroválvula en función del estado.

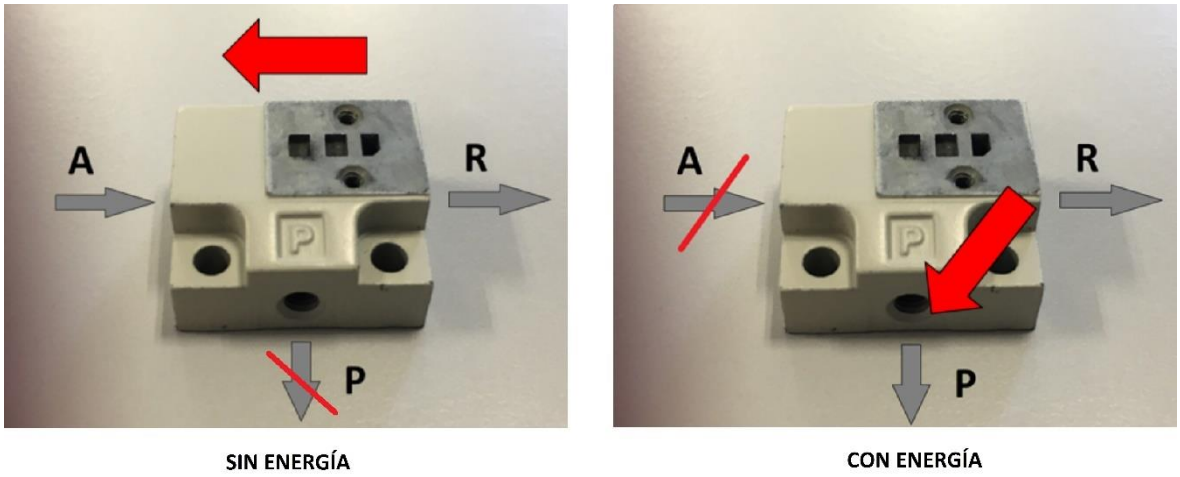


Figura 4. 9: *Base de la electroválvula. Conmutación entre las entradas y salidas para estados con o sin energía.*

Este tipo de conmutación entre puertas no es al ciento por ciento lo que necesitamos para crear el circuito de adquisición de odorante. Se pretende crear un array de electroválvulas interconectadas entre sí de manera consecutivas para lo que se ha decidido colocar las bases de montura según se muestra en la Figura 4. 10. Se realiza para ello con ayuda de una máquina Dremel y una broca de 3 mm de diámetro un agujero que comunique la apertura P con el punto rojo de la parte trasera de la electroválvula, al cual denominaremos C. Este agujero será necesario realizárselo a 3 de las 4 bases que se van a utilizar.

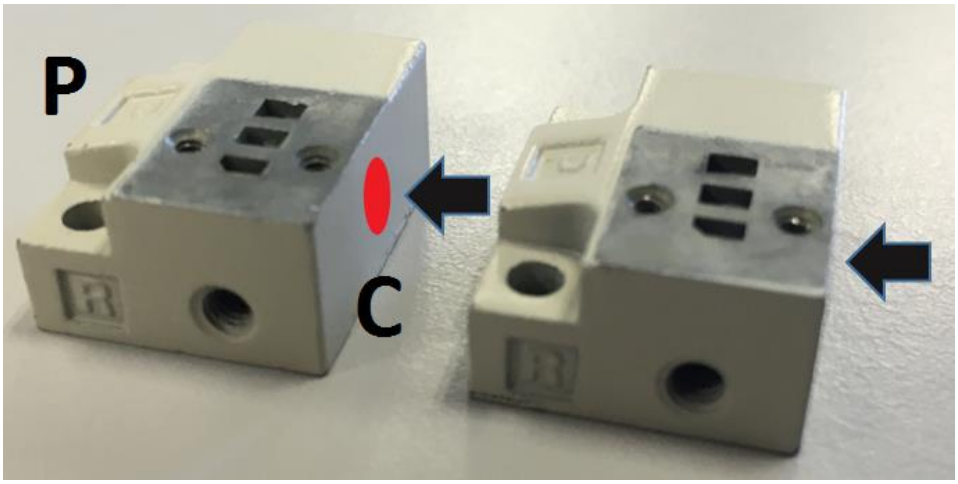


Figura 4. 10: *Perforación de la base de las electroválvulas.*

Una vez realizados los agujeros se procede a interconectar las bases de las electroválvulas, para ello se coloca la base no agujereada al principio, y consecutivamente, el resto uniendo el agujero P con el C de la siguiente base de montaje. Para la fijación se ha utilizado un pegamento de contacto. Una vez unidas se introducen en las puertas A y en la

última puerta P acoples de conexión para tubo de 3 mm. El array de electroválvulas queda como se muestra en la Figura 4. 11.

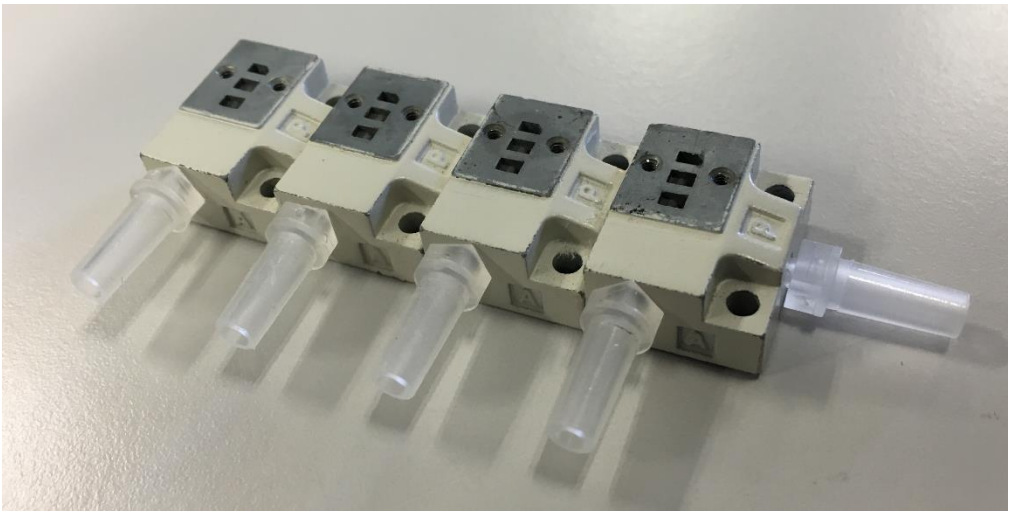
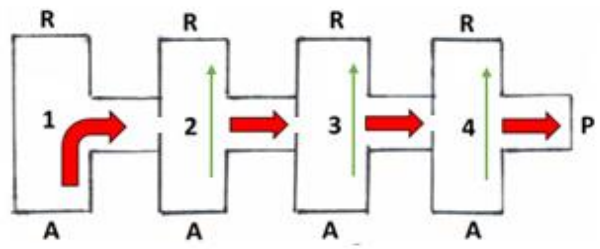
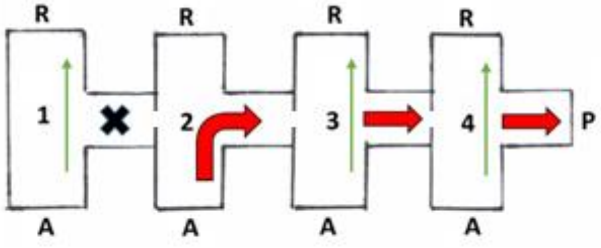


Figura 4. 11: *Array de bases de montaje para las electroválvulas.*

Con las modificaciones realizadas en las bases de montaje se consigue que al conmutar entre estado de energía o sin energía cada una de las electroválvulas se permita el paso del odorante deseado. En el cuadro que se muestra a continuación se puede observar los distintos modos de conmutación que permite este array.

Apertura electroválvula 1	**Apertura electrovávula 2**
	
Válvula 1 con energía, permite el paso desde A1 hasta P. Válvulas 2, 3 y 4 sin energía, se conmutan entre A y R cada una, haciendo que los odorantes conectados a estas no interfieran con el de la válvula 1.	Válvula 2 con energía, permite el paso desde A2 hasta P. Válvulas 1, 3 y 4 sin energía, se conmutan entre A y R cada una, haciendo que los odorantes conectados a estas no interfieran con el de la válvula 2.

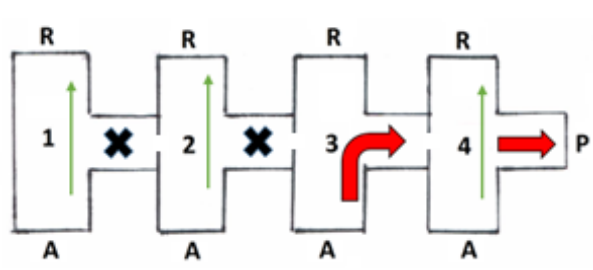
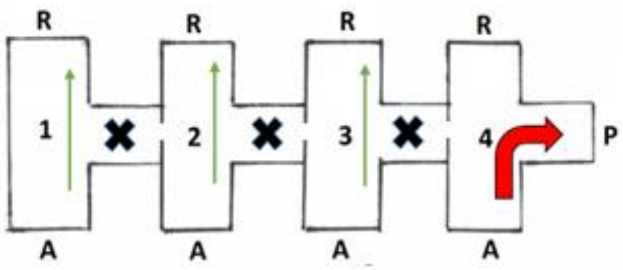
Apertura electroválvula 3	Apertura electroválvula 4
	
Válvula 3 con energía, permite el paso desde A3 hasta P. Válvulas 1, 2 y 4 sin energía, se conmutan entre A y R cada una, haciendo que los odorantes conectados a estas no interfieran con el de la válvula 3.	Válvula 4 con energía, permite el paso desde A4 hasta P. Válvulas 1, 2 y 3 sin energía, se conmutan entre A y R cada una, haciendo que los odorantes conectados a estas no interfieran con el de la válvula 4.

Tabla 7: *Esquema de conmutación entre electroválvulas.*

4.5.2. Conexión de los elementos del circuito de adquisición

En cada una de las puertas A de las electroválvulas se conecta un acople de conexión y a él un tubo de 3 mm en cuyo extremo se ha fijado una aguja. En la puerta P se conecta otro acople que ira unido a la caja hermética en la que se emplaza el sensor. Para la realización de esta caja se ha utilizado un tubo de plástico blanco e inerte de en torno a 1.5 cm de diámetro. Como se puede observar en la Figura 4. 12 el sensor se emplaza de forma que el aire con odorante procedente del array impacte de frente contra él. Un extremo del tubo estará cerrado y en el otro se realiza la conexión con el motor de succión, para ello se utilizar un acople y tubo de 6 mm de diámetro. La flecha azul indica la dirección de flujo del odorante.

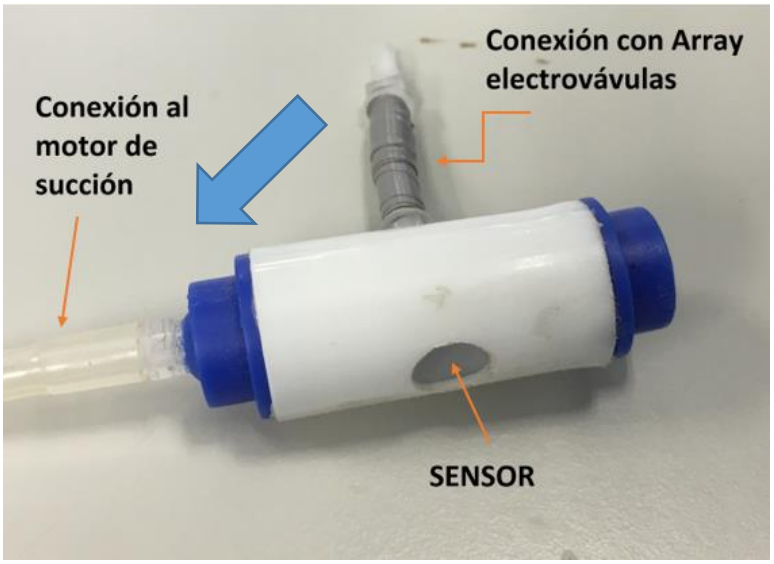


Figura 4. 12: *Caja hermética, emplazamiento del sensor TGS2600.*

El circuito de adquisición ya está casi listo, se realizan las uniones de sus componentes y se fijan con cinta aislante para asegurarnos de que no haya pérdidas. Se emplaza cada componente en su lugar sobre la base de madera y se utiliza alambre para su fijación. En la Figura 4. 13 se puede ver el circuito de adquisición de odorante emplaza en la plataforma de detección.

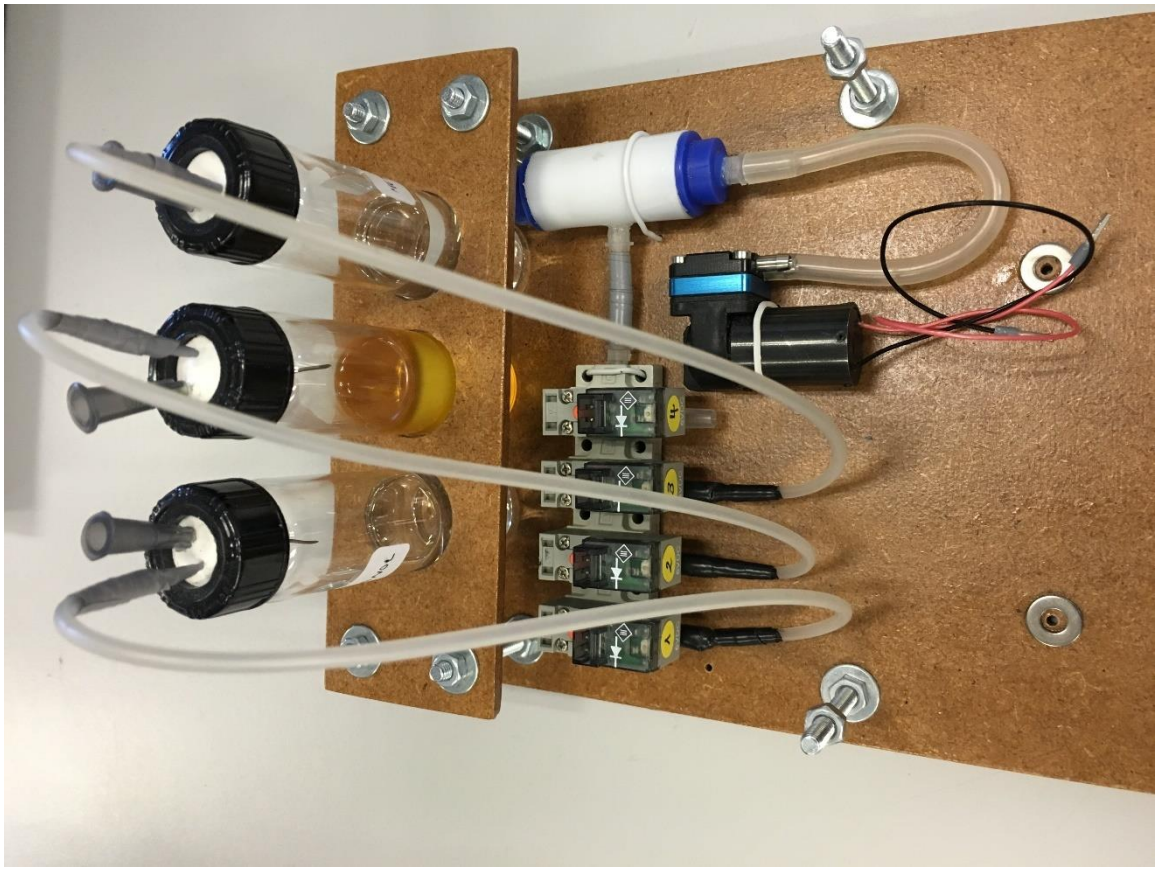


Figura 4. 13: *Circuito de adquisición de odorante montado en la plataforma.*

5. Desarrollo de la plataforma

5.1. Introducción

En esta sección se expondrá el diseño de la plataforma, el interconexionado de los componentes elegidos, así como, los circuitos de adaptación necesarios y más componentes secundarios.

Finalmente se ha optado por realizar tres circuitos para distintos tipos de experimentaciones. Se realizará un circuito para la adquisición de odorante con un sensor TGS2600 [10] sin realizar modulación, otro para realizar modulaciones en amplitud y un tercero para las modulaciones en frecuencia.

Hay que entender primero sobre qué parámetro se realizan las modulaciones, aunque ya se indicó en anteriores secciones. Se trata de la temperatura de calentamiento del sensor, este presenta unas características de salida distintas en función del valor de esta componente. Por lo que, será nuestro variable para la modulación, bien sea en amplitud o frecuencia. Y en caso de no realizar modulación se fijará en su valor máximo (5V).

5.2. Hardware

5.2.1. Circuito TGS2600 sin modulaciones

Como ya se introdujo en las secciones [3.3] el sensor utilizado para el desarrollo de la plataforma es el TGS2600 de Figaro. Este sensor a la vista de la Figura 5. 1 internamente está compuesto por un circuito con dos resistencias, denominadas resistencia interna del sensor y resistencia de calentamiento, y dos voltajes de alimentación. Estos son voltajes son, uno fijo, denominado voltaje de alimentación, y otro, que puede ser variable, denominado voltaje de calentamiento o temperatura de calentamiento del sensor.

Uno de los objetivos de este proyecto era introducir técnicas de modulación en la plataforma para adquisición de odorante, dichas modulaciones se realizan sobre la temperatura de calentamiento. A parte de las experimentaciones con modulación, se ha creído conveniente introducir un sensor más para poder realizar experimentaciones con odorantes sin hacer uso de modulaciones para el control de su temperatura de calentamiento. En esta sección se expondrá el circuito necesario para realizar la medida del sensor.

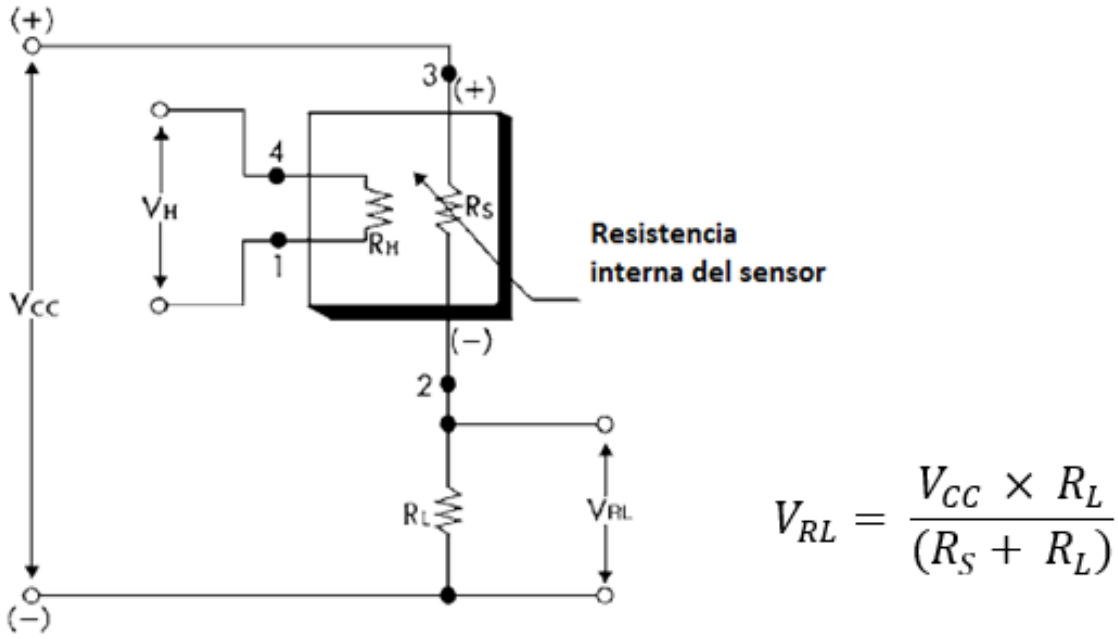


Figura 5. 1: *Circuito de acondicionamiento del sensor TGS2600 y cálculo de la tensión en la resistencia de carga. Imagen modificada de [10].*

Como no se va a realizar ningún tipo de modulación, la temperatura de calentamiento de fijará en 5V (aunque es un parámetro que en alguna experimentación igual conviene fijarlo en otro valor), y el circuito de medida será un divisor de tensión a la salida del sensor. En la Figura 5. 1 se puede observar el sistema de medida a implementar y una ecuación con el valor de la tensión en la resistencia de carga. Esta ecuación se ha calculado a partir de la información del datasheet del componente [10], en donde nos especifican que el valor en cada instante de la resistencia interna del sensor viene dado por:

$$R_S = \frac{V_{CC} \times R_L}{V_{RL}} - R_L$$

Esta ecuación de R_S viene de realizar el calculo del circuito divisor de tensión. Como la medida será tomada por un puerto ADC de la Beaglebone, y tal y como se indicó en la sección [3.2.3], estos pines están limitados a 1.8V el valor de la resistencia de carga, R_L, vendrá impuesto por esta limitación. Además, de esta limitación hay que tener en cuenta los rangos de variación de la resistencia interna del sensor, para que en ningún momento la caída de tensión sobre $R_L$supere los 1.8V. A continuación, se muestran los cálculos realizados:

1. Haciendo uso del datasheet del componente (TGS2600), sabemos que la resistencia del sensor varia de 10k-90kΩ en el aire, aproximadamente. Esta variación se produce sin presencia de odorante, para presencia de odorante utilizamos la gráfica de sensibilidad (véase Figura 3. 12), donde se puede apreciar que en presencia de Hidrógeno la resistencia varia en un factor de 0.1, por lo cual el rango de variación máximo será de 1k-90kΩ.

2. Se despeja de la ecuación anterior el término R_L:

$$R_L = \frac{V_{RL} \times R_S}{V_{CC} - V_{RL}}$$

3. Se calcula el valor de R_L que certifique no sobre pasar los 1.8 V en la resistencia de carga para el valor mínimo de la resistencia del sensor, $R_{Smin} = 1k$.

$$R_L = \frac{V_{RL} \times R_S}{V_{CC} - V_{RL}} = \frac{1.8V \times 1k}{5 - 1.8\,V} = 562.5\ \Omega$$

4. Se calcula la tensión que cae en $R_L = 562.5\ \Omega$ para el valor mínimo de la resistencia del sensor, $R_{Smax} = 90k$.

$$V_{RL} = \frac{V_{CC} \times R_L}{(R_S + R_L)} = \frac{5V \times 562.5\ \Omega}{(90k + 562.5\ \Omega)} = 0.031\,V$$

5. Con lo cual, la resistencia que certifica que no se sobre pasen los 1.8 V de tensión en la resistencia de carga (que va ser medida por la BBB) es $R_L = 562.5\ \Omega$. El límite inferior de tensión en 0.031 V y el superior 1.8 V.

6. Para no trabajar justo en el límite superior con 1.8 V y asegurarnos de que en ningún momento a la placa pueda llegar una tensión superior, se decide aplicar un pequeño margen y elegir una resistencia de carga algo menor. Se utilizará una resistencia de carga, $R_L = 440\ \Omega$, en la que caerá una tensión máxima y mínima de:

$$V_{RLmax} = \frac{V_{CC} \times R_L}{(R_{Smin} + R_L)} = \frac{5V \times 440\ \Omega}{(1k + 440\ \Omega)} = 1.528\,V$$

$$V_{RLmin} = \frac{V_{CC} \times R_L}{(R_{Smax} + R_L)} = \frac{5V \times 440\ \Omega}{(90k + 440\ \Omega)} = 0.0243\,V$$

5.2.2. Circuito modulación en Frecuencia

Como se indicó en la sección [3.5] el circuito de auto-modulación de temperatura propuesto por Martinelli se centra en el uso de un multivibrador del tipo 555. En nuestro caso se ha elegido un NE555 [17]. Este dispositivo permite ser configurado en dos modos distintos (véase Anexo [C]), siendo su modo astable el que nos resulta de interés.

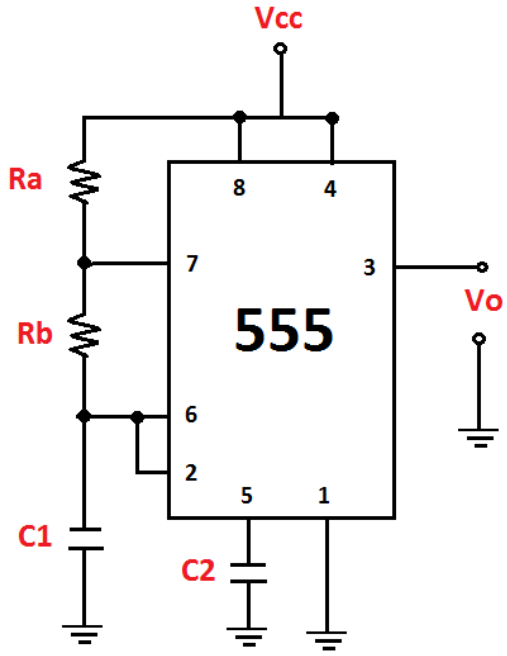


Figura 5. 2: *Oscilador 555 configuración astable.*

La configuración astable (Figura 5. 2) proporciona dos estados, alto y bajo, cuya duración viene dada por los valores de Ra, Rb y C1. Estos tiempos van a depender del valor de los componentes externos del circuito. Analizando el circuito de la figura se obtienen las siguientes ecuaciones para los tiempos de encendido y apagado.

Figura 5. 3: *Estados del oscilador 555.*

$$t_{ON} = 0.693 \times (R_A + R_B) \times C1$$

$$t_{OFF} = 0.693 \times R_B \times C1$$

Donde t_{ON}, se refiere al tiempo de encendido y corresponde al tiempo de carga del condensador C. Y donde t_{OFF}, se refiere al tiempo de apagado y corresponde al tiempo de descarga del condensador C.

El tiempo total vendrá dado por: $$T = t_{ON} + t_{OFF} = 0.693 \times (R_A + 2R_B) \times C1$$

Luego si se quiere obtener $t_{ON} = t_{OFF}$, implica que $R_B \gg R_A$

$$T = 1.38 \times R_B \times C1$$

Donde R_B va ha ser la resistencia interna del sensor. Teniendo en cuenta la hoja de características del sensor TGS2600 [10], sabemos que en el caso ideal esta resistencia variará desde aproximadamente 10kΩ hasta aproximadamente 90kΩ en el aire. Haciendo uso de la Figura 3. 12 donde se observa que la sensibilidad del sensor en el peor de los casos (en presencia de Hidrogeno) disminuye un orden de magnitud. Concluimos que la resistencia de sensor tiene un rango de variación aproximado de entre 1kΩ y 90kΩ.

La elección de los componentes externos del 555 se verá condicionada por el rango de valores que puede tomar la resistencia del sensor. De la hoja de características se sabe que el valor de la resistencia del sensor viene determinado por la siguiente formula:

$$R_s = \frac{V_c \times R_L}{V_{out}} - R_L$$

Haciendo uso de la herramienta Matlab se ha simulado el valor de la resistencia del sensor para distintos valores de resistencias de carga, con el fin de ver como varia la salida del sensor. Y con ello poder decidir cuál es la resistencia que proporción una salida más lineal.

El código utilizado se puede ver en el Anexo [G]. Como resultado se han obtenido las siguientes gráficas. De izquierda a derecha y de arriba abajo, se muestra la salida del sensor en función de la variación de la resistencia interna del sensor para un valor fijado de resistencia fija de carga. El valor más arriba a la izquierda corresponde con RL = 1kΩ, el siguiente RL = 10kΩ y el resto van incrementándose en 10 kΩ hasta RL = 90kΩ.

Observando las gráficas de la Figura 5. 4 se ha decidido que la resistencia utilizar sea de en torno a 30 kΩ, puesto que genera la salida con mayor rango de voltaje y aproximadamente lineal. Puesto que en el laboratorio contábamos con resistencias de 27 kΩ, esta ha sido la elegida con resistencia Ra.

Con la resistencia de carga ya elegida, la cual se asignará a Ra, se procede al cálculo del valor del condensador. En este punto se tiene en cuenta que en el peor de los casos la duración de un pulso completo a la salida del 555 debe ser lo suficientemente larga como para poder ser medida por la BBB.

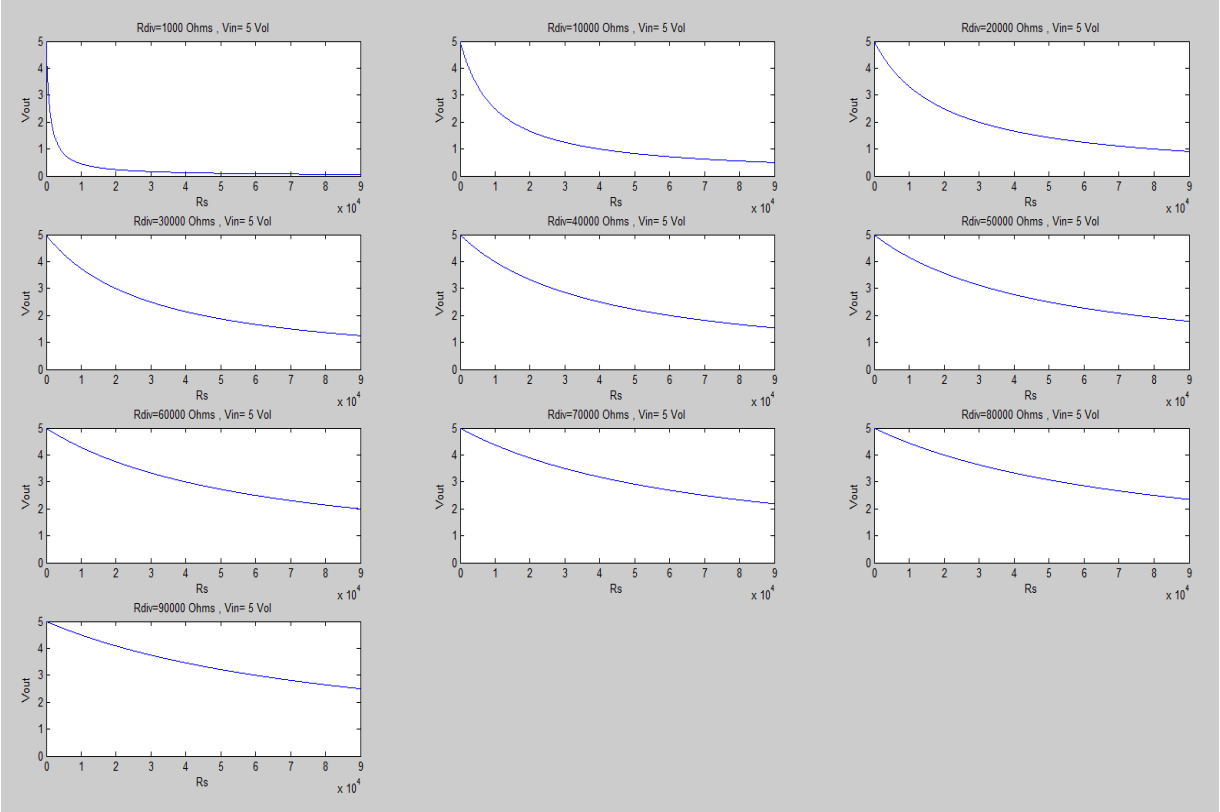


Figura 5. 4: *Gráficas Salida del sensor. Se representa en función de la variación de la resistencia del sensor y una resistencia de carga fija.*

El peor de los casos ocurrirá cuando en presencia de odorante el sensor disminuya su resistencia al máximo y con ello se producirá un aumento de la frecuencia de oscilación del 555.

$$T_{min} = 0.693 \times (27\text{k}\Omega + 2 \text{ x } 1\text{k}\Omega) \times 100\mu\text{F} = 2.0097 \text{ seg}$$

Y el caso opuesto, con frecuencia mínima de oscilación la duración del pulso completo será de:

$$T_{max} = 0.693 \times (27\text{k}\Omega + 2 \text{ x } 90\text{k}\Omega) \times 100\mu\text{F} = 14.345 \text{ seg}$$

Con esto, ya quedan elegidos los componentes externos necesarios para el circuito. Se utilizarán una resistencia Ra = 27k**Ω**, Rb = Resistencia sensor, C1 = 100 µF y C2 = 0.01 µF. La tensión de alimentación la marca el TGS2600 y será de 5V, su tensión de funcionamiento. El NE55 trabaja de 4.5 V a 16 V por lo que 5 V está ente sus tensiones de alimentación.

Finalmente, para eliminar posibles ruidos y picos en la señal de salida se ha decido introducir un condensador de desacoplo entre alimentación y tierra de 0.01 µF, y una resistencia de carga entre alimentación y la salida del 555 de 1 k**Ω**. Quedando el circuito multivibrador astable con el sensor integrado como se muestra en la Figura 5. 5. Donde se muestra el esquema de conexión del circuito completo, con las entradas y salidas

correspondientes a la BBB. Para el correcto funcionamiento de este circuito, nótese que funciona a 5V, será necesario alimentar la BBB con el trasformador de corriente.

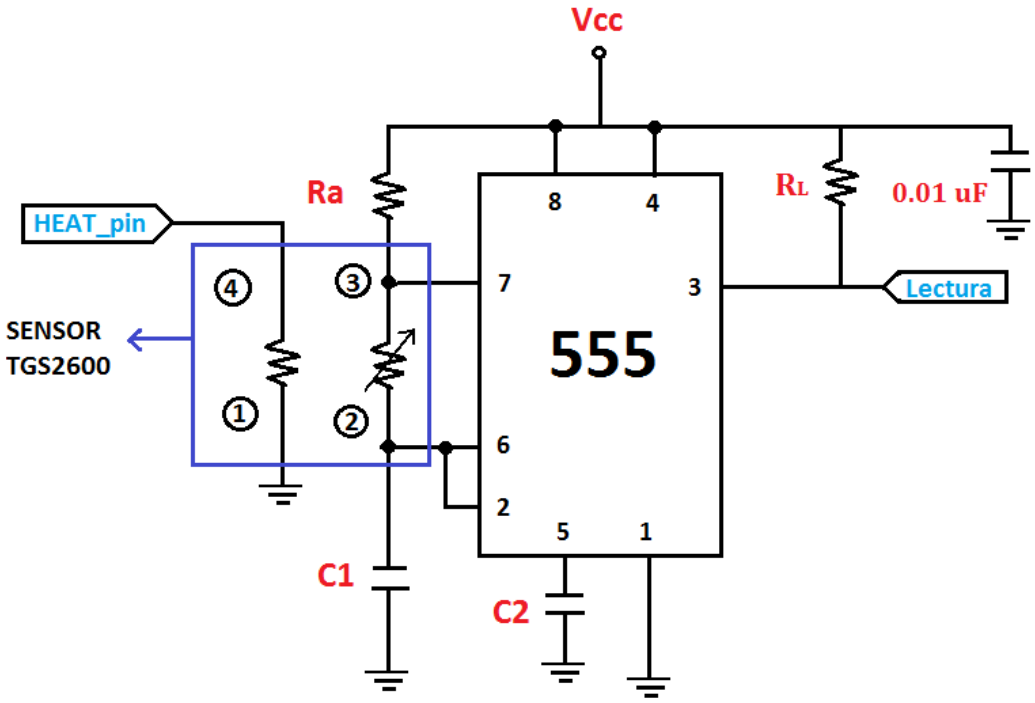


Figura 5. 5: *Circuito multivibrador con Sensor TGS2600 y NE555. Figura adaptada de [9].*

Donde 1 corresponde con Rh (-), 2 con Rs (-), 3 con Rs (+) y 4 corresponde con Rh (+).

Teóricamente a la salida del 555 obtendremos una señal cuadrada cuyo valor máximo será de entorno al 70 % del valor de alimentación. Esto quiere decir que la señal de salida tendrá un valor de unos 3.5 V. Como vimos en la sección [3.2.3]**¡Error! No se encuentra el origen de la referencia.**, la BBB tiene ciertas limitaciones que hay que tener en cuenta cuando se utilizan sus pines de entrada/salida. Puesto que, como se verá en la sección [5.2.2], el pin utilizado para la lectura es un GPIO y este tiene su máximo en 3.3 V, es necesario realizar un circuito de adaptación antes de conectar con la BBB.

Para limitar la tensión existen distintas opciones, desde un simple divisor de tensión hasta un puente de diodos, pasando por la utilización de transistores. En el caso de este proyecto se ha decidido que lo mejor es utilizar transistores, en concreto transistores Darlington.

El transistor Darlington se puede encontrar a la venta en distintos formatos, en este proyecto se ha decido utilizar un integrado que contiene 4 transistores Darlington en su interior. Se trata del ULN2003 [32].

Este componente nos permite limitar la tensión al voltaje que deseamos. Se utiliza uno de los siete transistores Darlington que contiene. Se aplica sobre la entra COM un voltaje de

alimentación, que será el voltaje al que el integrado pondrá la salida 1C. De esta forma fijando el voltaje de alimentación de este componente en 3.3 V (utilizando directamente la salida del voltaje que proporciona BBB) nos aseguramos que tanto la corriente como la tensión de entrada en la BBB no supere los máximo impuestos por las limitaciones de la misma. El puerto 1B del componente se conectará a la salida del 555. Cuando por esta entrada se detecte un valor alto, el Darlington actuará y pondrá la salida 1C al valor de 3.3 voltios.

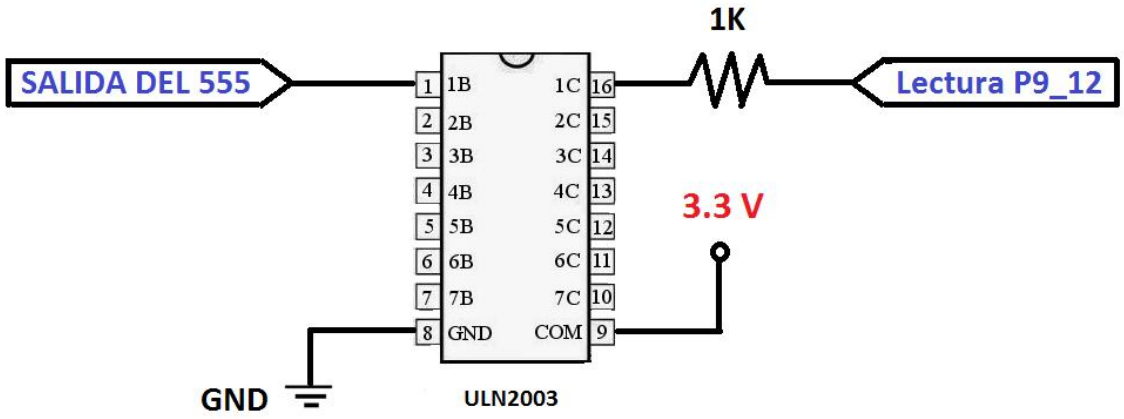


Figura 5. 6: *Circuito integrado ULN2003 [32].Limitador de tensión.*

Con el fin de asegurar que no se produzcan corriente de entrada no deseadas en la BeagleBone

En la Figura 5. 7 se puede observar el circuito montado en la protoboard de la plataforma. En la sección 5.3.3 se detallarán los puertos de la BBB que se utilizan para la modulación, así como el funcionamiento del código.

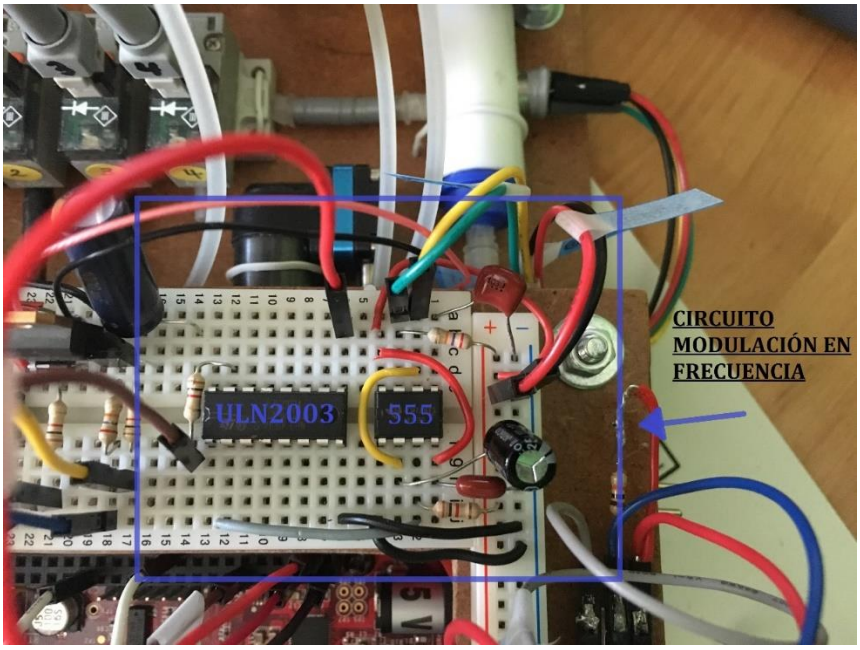


Figura 5. 7: *Circuito de modulación en frecuencia montado en la protoboard de la plataforma.*

5.2.3. Circuito modulación Amplitud

Para realizar la lectura de odorante utilizando la modulación de temperatura el circuito es distinto. No vale la adaptación realizada en la sección anterior, sino que habrá que realizar un nuevo circuito de adaptación y como consecuencia utilizar otro sensor TGS2600 para no tener que conectar y desconectar el sensor cada vez que se quiere utilizar una modulación u otra.

El circuito que se ha diseñado parte del circuito diseñado por Alejandro Pequeño en su proyecto final de carrera [7]. Al que se le han realizado algunas variaciones para adaptarlo al uso del sensor TGS2600 (Alejandro utilizaba el TGS2611) y al igual que con el circuito de la sección [5.2.2] será necesario limitar la tensión de salida para no quemar la BBB. En este caso se optó por utilizar un divisor de tensión.

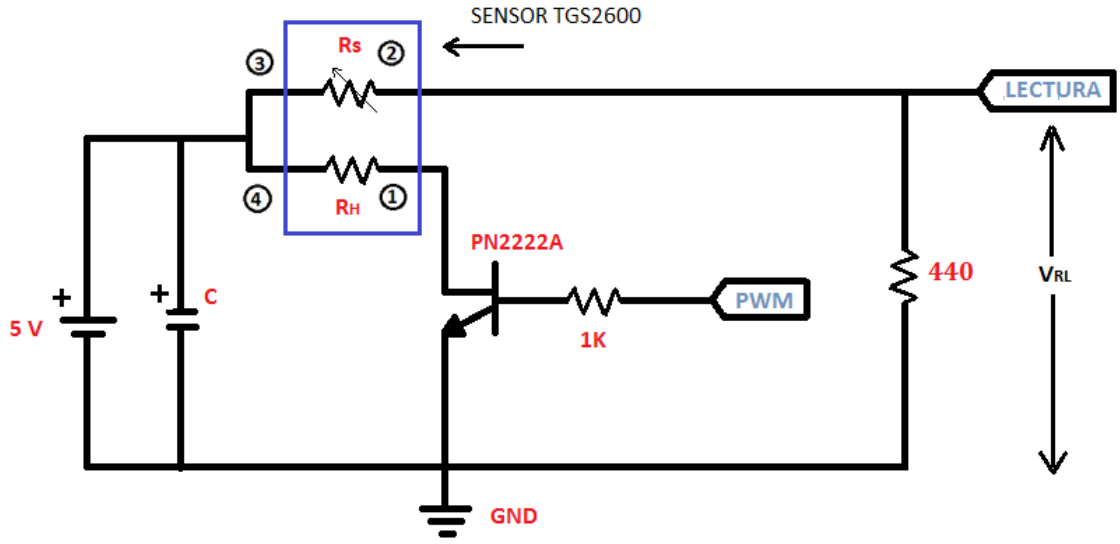


Figura 5. 8: *Circuito de adaptación para medida ADC del sensor TGS2600. Circuito adaptado del circuito de control de temperatura de [7].*

Donde 1 corresponde con Rh (-), 2 con Rs (-), 3 con Rs (+) y 4 corresponde con Rh (+).

El transistor del circuito sirve para ampliar el rango dinámico del calefactor hasta los 5 voltios que es la temperatura de calentamiento máxima (Vh). El control de la base del transistor se realiza mediante el pin del PWM de la BBB, limitado en tensión a 3.3 V. Para tener una relación directa entre PWM y calefactor a la vez que exista un punto de corte coincidente a 0 voltios, elegimos un transistor NPN. Entre el transistor y la salida PWM se coloca una resistencia de protección para evitar posibles corrientes inversas que entren en la BBB. Al aplicar 0 voltios en la salida PWM el transistor no permitirá el paso de corriente del emisor al colector. Al aplicar voltaje por el puerto PWM, en la base del transistor, este

dejará pasar la corriente a su través y creará una diferencia de voltaje entre el emisor y el colector (voltaje de calentamiento) al aplicar 3.3 voltios en la salida PWM.

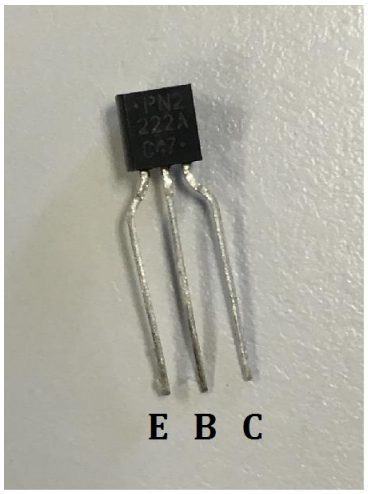
El cálculo de las corrientes del colector está determinado por la alimentación del sistema y la resistencia de calefacción del sensor, dado que alcanza la corriente máxima cuando el sistema consume 5 voltios de alimentación. Con lo cual, según las ecuaciones de control del transistor, el valor de amplificador de entrada:

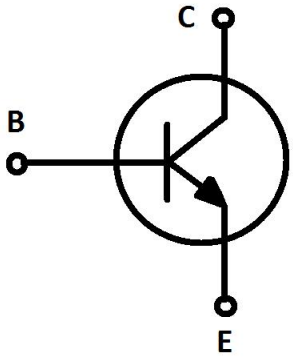
$$I_C = {}^{5v}/_{R_H} \qquad \beta = {}^{I_c}/_{I_B}$$

$$I_B = 6\,mA\ máximo\ del\ pin\ PWM\ (BBB)$$

$$\beta = 10$$

Tenemos un valor de amplificación muy pequeño, por lo que no es un parámetro importante en la elección del transistor. De entre los transistores del mercado se decidió utilizar un PN2222A [29] que cumple con las especificaciones del sistema.





B - base
C - colector
E - emisor

Figura 5. 9: *Transistor PN2222A. Circuito interno y pines de conexión del encapsulado.*

El condensador utilizado trabaja a modo de bypass para eliminar el ruido ocasionado por la rama de calefacción del sistema conectada a la alimentación que a su vez está conectada con la rama de medida del divisor de tensión del sensor, y, por lo tanto, puede afectar en

gran medida a las medidas del sistema. Referenciando la guía de filtrado [31], se selecciona un condensador de 1000 μF.

La resistencia de divisor de tensión sobre la que se efectúa la medida es de 440 kΩ de acuerdo con lo citado en la sección [5.2.1] donde se explica la elección de la resistencia de carga óptima. En la sección [5.3.4] se detallarán los puertos de la BBB que se utilizan para la modulación, así como el funcionamiento del código.

5.2.4. Circuito control electroválvula

Como se indicó en la sección 3.6 y 4.5, se han utilizado cuatro electroválvulas SY114-VLOZ-Q para la realización del circuito de adquisición de odorante. Como se puede ver en su datasheet [18] estas electroválvulas trabajan a 6 V. La BBB como máximo nos puede ofrecer 5 V por lo que es necesario amplificar este voltaje o utilizar una fuente de alimentación de 6 V.

Para no sobrecargar la BBB, y no exigirla un consumo excesivo de potencia se ha decido utilizar una fuente de alimentación para las electroválvulas. Se trata de una fuente de 6 V y 3 A. Viene con varias clavijas de conexión. Como se puede ver en la Figura 5. 10, una de las clavijas se ha fijado en la plataforma para facilitar la conexión.

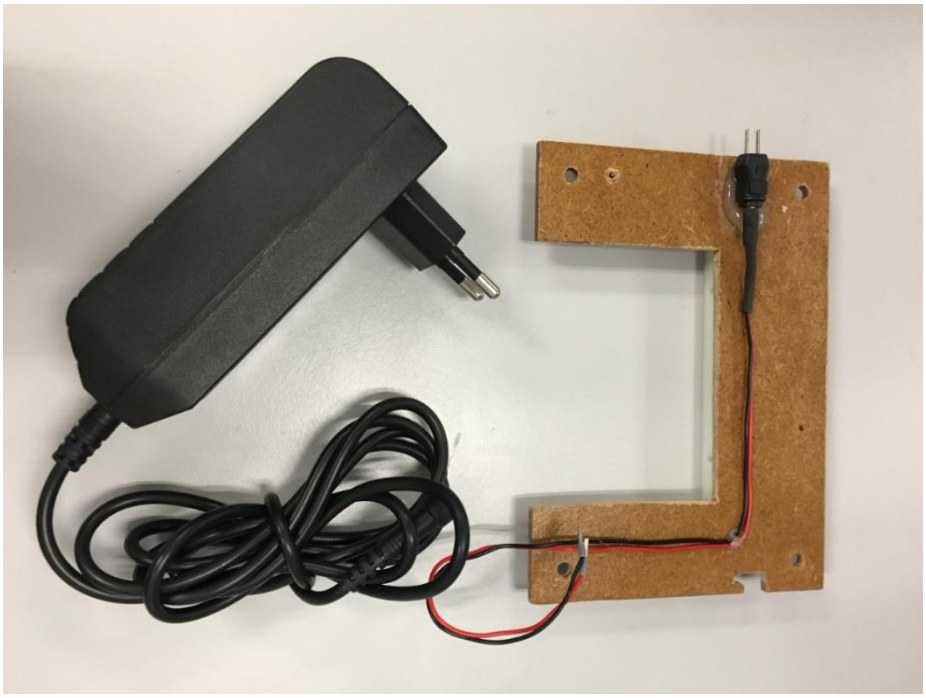


Figura 5. 10: *Fuente de alimentación de 6 voltios y 3 amperios para la alimentación de las electroválvulas. Integración del conector en la plataforma.*

El control de las electroválvulas se hace a través un chip integrado ULN2003 [32] que contiene un array de transistores Darlington. En concreto tiene 7 pares de transistores NPN

que permiten controlar componentes de alta tensión. Es un elemento comúnmente utilizado para la conmutación. En la Figura 5. 11 podemos ver el esquema interno del encapsulado.

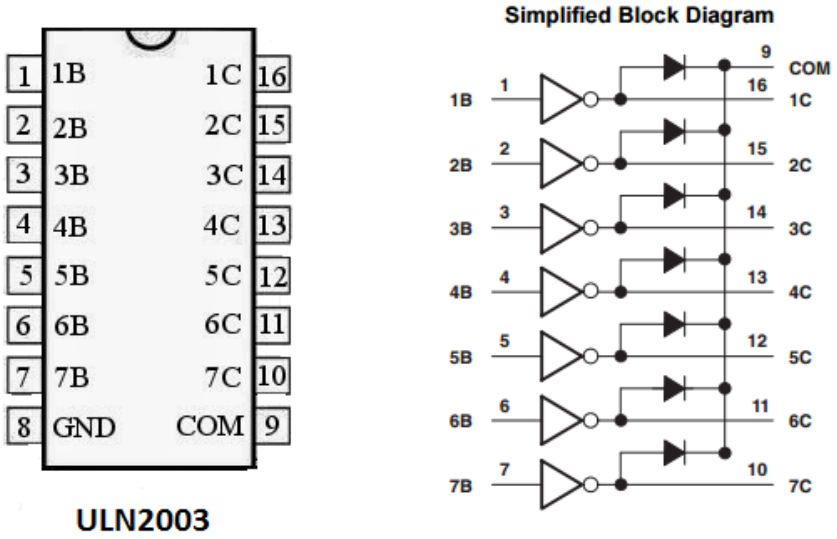


Figura 5. 11: *Chip integrado ULN2003. Array de 7 transistores Darlington. Pines del encapsulado y esquema de conexión interno.*

El ULN2003 funciona a modo de conmutador. Se alimenta a 6 voltios por el pin 9 (COM) con la señal proveniente de la fuente de alimentación de 6 V. El pin 8 (GND) se conecta a la tierra de la misma fuente. Cada una de las entradas del integrado (1B, 2B, 3B y 4B) se conectan a una resistencia de protección de 1kΩ y de esta a los puertos de salida de la BBB destinados al control de las electroválvulas. Las salidas del integrado (1C, 2C, 3C y 4C) se conectan al polo negativo de cada electroválvula, y el polo positivo de las válvulas va a 6 voltios.

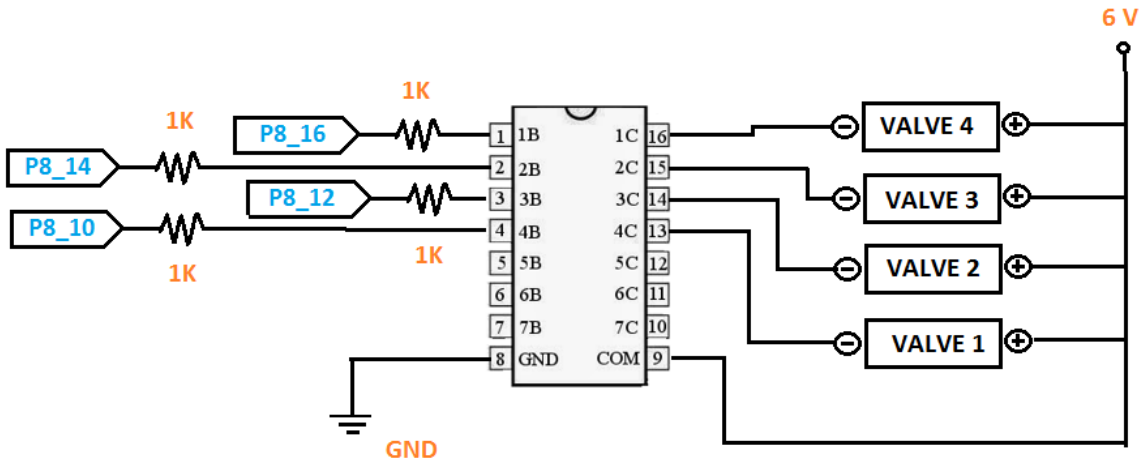


Figura 5. 12: *Circuito de control de electroválvulas basado en ULN2003.*

Cuando la BBB envía un pulso alto por una de las salidas de control, el ULN2003 conmuta y permite el paso de la corriente por la electroválvula seleccionada, conectando la tierra a la pata correspondiente (1C, 2C, 3C y 4C). En la siguiente Figura 5. 12 se presenta un esquema de la conexión realizada.

Los pines P8_10, P8_12, P8_14 y P8_16 se configuran como GPIO de salida y se ponen a 1 cuando se quiere permitir el paso de corriente a través de la válvula. En el código explicado en la sección [5.3] se detalla la configuración y conmutación de los puertos para el correcto funcionamiento del circuito.

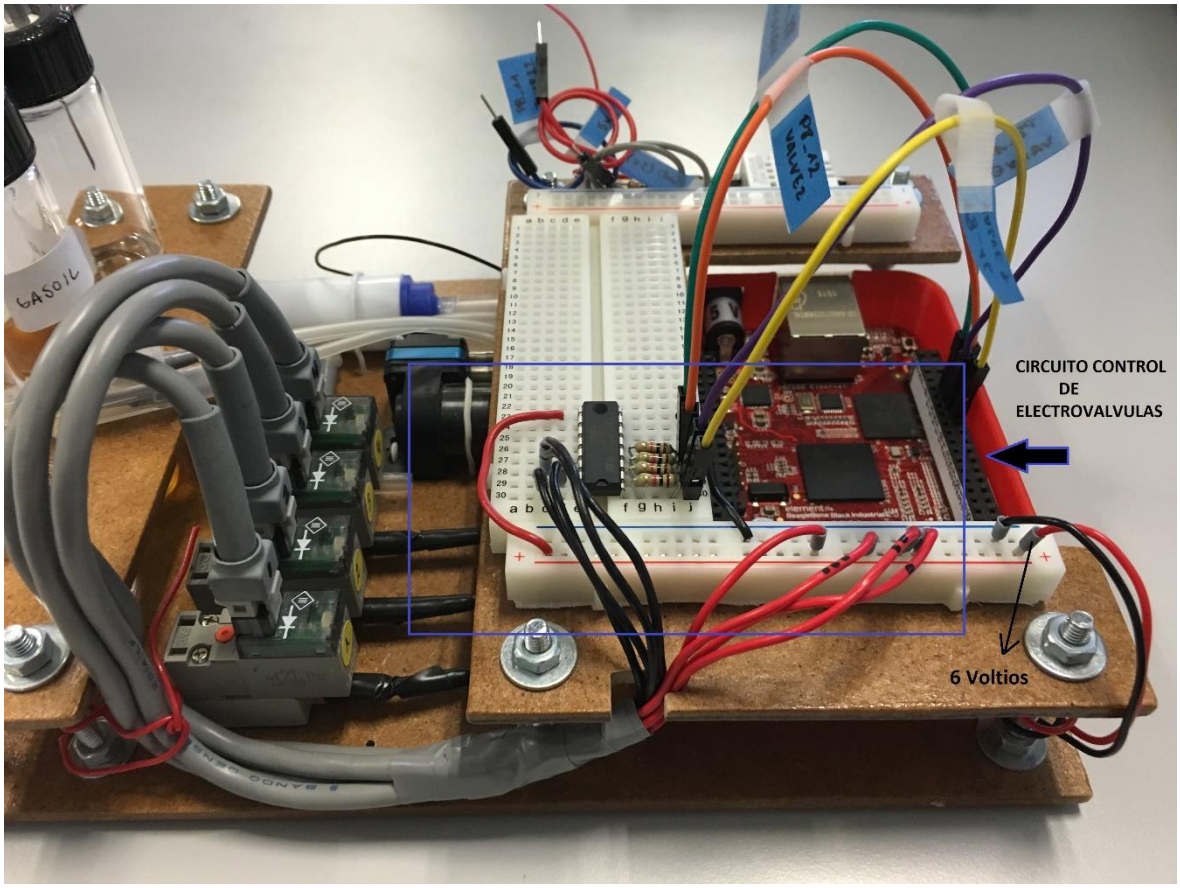


Figura 5. 13: *Circuito control de electroválvulas conectado en la protoboard de la plataforma.*

5.2.5. Circuito potencia motor

Para llevar a cabo la succión del odorante se ha elegido, como se explicó en la sección [3.6.1], un RS D200 Micropump. Dicho motor funciona a distinta potencia de succión en función de su tensión de alimentación que puede variar de 3 a 4.5 voltios.

Dado que la BBB no tiene puertos PWM que permitan variar su valor entre los voltajes de trabajo del motor, es necesario crear un circuito de adaptación. La salida PWM de la BBB aplica un voltaje máximo de 3.3 V y mínimo de 0 V, y su resolución es de 12 bits. Por consiguiente, es necesario un circuito amplificador de voltaje.

Puesto que se tiene una fuente de alimentación de 6 voltios integrada en la plataforma, se decide hacer uso de ella para proporcionar energía al motor. Para ello se utiliza un TIP120 [33], se trata de un transistor que básicamente funciona como amplificador o como un interruptor electrónico.

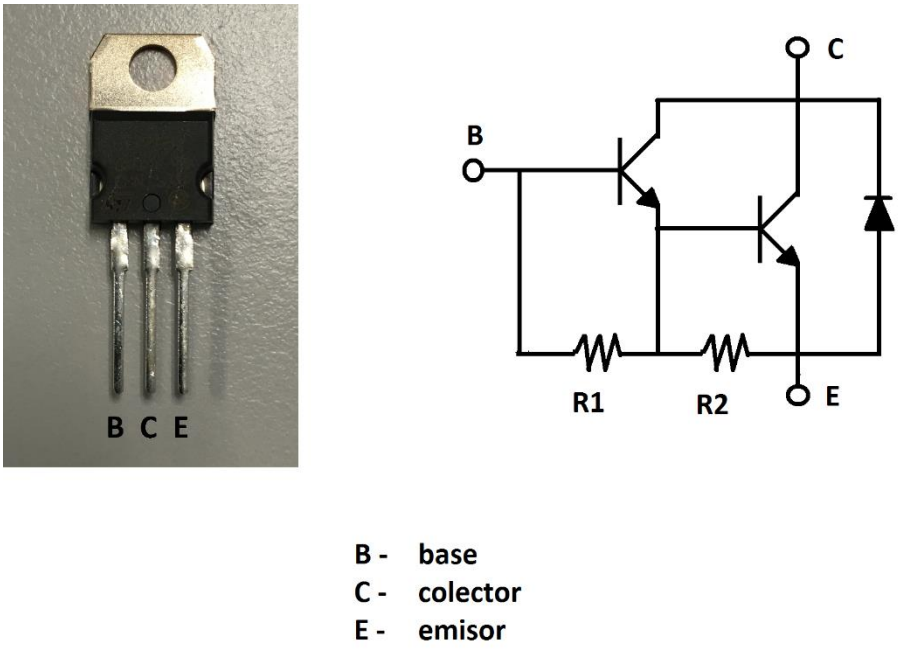


Figura 5. 14: *TIP120 [33]. Circuito interno y pines de conexión del encapsulado. Imagen adaptada del datasheet [33].*

El TIP120 posee una entrada denominada Colector, una salida llamada Emisor, y un control denominado Base. Cuando se envía una señal de alto a la base, el transistor cambia y permite que la corriente fluya desde el colector para el emisor. Esta señal será suministrada por el puerto PWM de la BBB y en función de su valor, dejará pasar más o menos corriente entre colector y emisor (véase Figura 5. 15).

Al conectar la alimentación al motor se produce un pico negativo de la tensión, que puede dañar la placa. Será necesario entonces la utilización de un diodo como rectificador. Éste actuará como una válvula unidireccional que sólo permite que la corriente fluya en una dirección, por lo que el circuito estará protegido por si la alimentación del motor produce un pico de corriente o si el motor consume demasiada. Se utiliza un diodo 1N4001 [36], es importe colocarlo en la posición correcta. Por último, se utiliza también una resistencia de 1kΩ para proteger a la base del transistor de un exceso de corriente que podría dañarlo y a la entrada de la BBB.

En la Figura 5. 15 se presenta el esquema del circuito realizado:

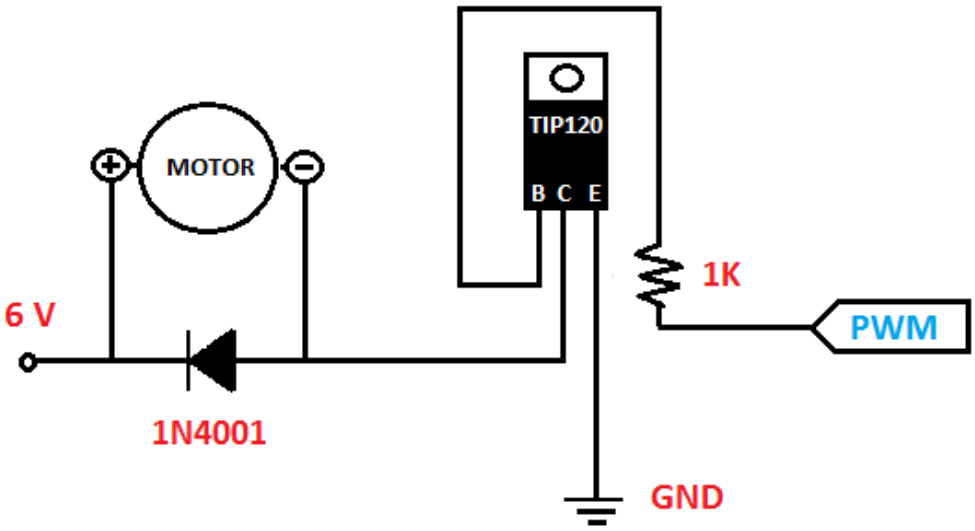


Figura 5. 15: *Circuito control de potencia motor succión.*

Con este circuito conseguimos que al variar el rango de trabajo del PWM (pin P9_21) entre el 50 y 100% de su valor máximo la tensión que cae al motor varíe de 3 a 4.5 voltios, y por consiguiente permite utilizar distintas potencias de succión. Nótese que se utiliza una resistencia entre el circuito y el PWM para evitar posibles entradas de corriente en la placa.

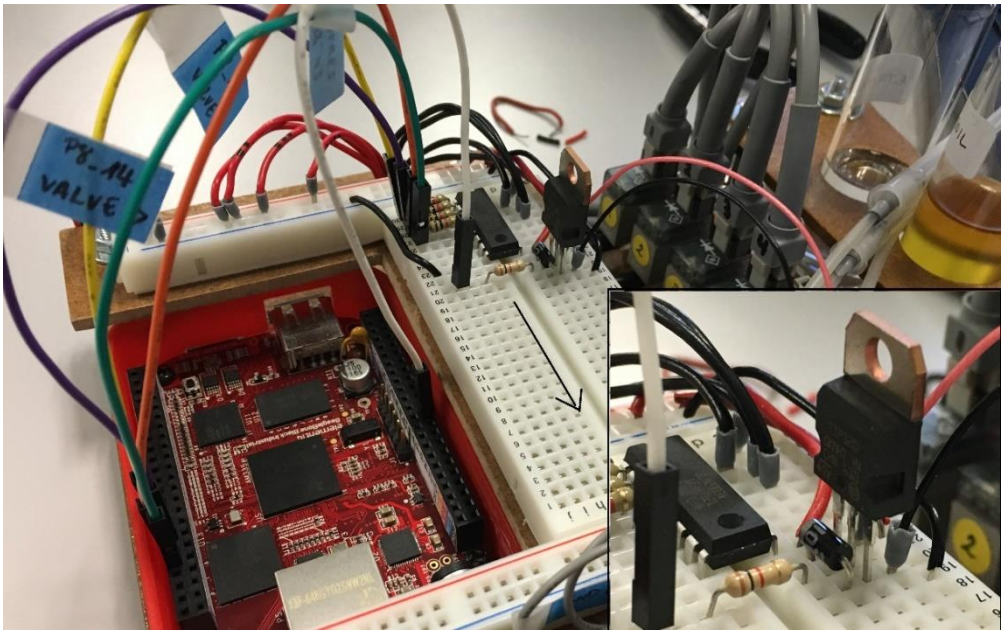


Figura 5. 16: *Circuito control motor de succión montado en la protoboard de la plataforma.*

En la Figura 5. 16 se puede observar el circuito diseña para el control del motor de succión, instalado en la protoboard de la plataforma de experimentación.

5.3. Software de control de la plataforma

La presente sección explica el código realizado para el control de la plataforma de odorante. El código utilizado es Python y se ha hecho uso de librerías de terceros, que se pueden encontrar en formato libre por internet. En concreto se han utilizado la librería de Adafruit para el control de GPIOs y la librería de Adafruit para la lectura de temperatura y humedad. Se explicará su funcionamiento, aunque también se puede encontrar información sobre su uso en el Anexo [E.8.1].

Se han realizado tres códigos de experimentación, que se han denominado Modulación Frecuencia, Modulación Amplitud y Limpieza. El primero utiliza el circuito realizado en la sección [5.2.2], el segundo el realizado en la sección [5.2.3] y el último realiza limpieza de válvulas. Aunque se trata de códigos de experimentación distintos, los tres tienen funciones o partes del código compartidas o muy parecidas, como pueden ser las que se encargan del control de las electroválvulas, la lectura de temperatura y humedad, etc. Se pueden ver los códigos completos en el Anexo [0]. A continuación, se explicarán las partes fundamentales del código.

5.3.1. Asignación de puertos y variables globales

Hay ciertas variables y puertos que son comunes para todas las experimentaciones.

En la siguiente tabla se muestra la asignación de puertos:

PUERTO	NOMBRE	COMANDO	COMENTARIO
P8_10	electrovalve1	`electrovalve1 = 'P8_10'`	Válvula 1 (METANOL)
P8_12	electrovalve2	`electrovalve2 = 'P8_12'`	Válvula 2 (ETANOL)
P8_14	electrovalve3	`electrovalve3 = 'P8_12'`	Válvula 3 (BUTANOL)
P8_16	electrovalve4	`electrovalve4 = 'P8_14'`	Válvula 4 (AIRE)
P9_21	motorPin	`motorPin = 'P9_21'`	Motor de succión
P8_11	Temp22	`Temp22 = 'P8_11'`	Sensor de temperatura/humedad

Tabla 8: *Asignación de puertos globales de la plataforma.*

En la Tabla 9 se muestran las variables globales del sistema.

VARIABLES	VALOR	COMANDO	COMENTARIO
x		x=[]	Almacena el número de muestra
tempTGS2600		tempTGS2600=[]	Almacena las temperaturas de calentamiento del sensor
Rl_2600	27000	Rl_TGS266=27000	Valor de la resistencia de carga
Vc	5	Vc=[]	Tensión de alimentación
SLEEP_tyh	59	SLEEP_tyh=59	Tiempo de espera entre capturas de temperatura/humedad

Tabla 9: *Variables globales del sistema.*

Dentro de los códigos de experimentación hay algunas funciones comunes, en concreto, las que se encargan de la apertura de las electroválvulas, arranque del motor y lectura de temperatura y humedad.

5.3.1.1. Función de apertura de las electroválvulas.

El código de la función es el siguiente:

```
# Apertura electrovalvula.

def apertura(electrovalvula):
     if electrovalvula == 1:
          GPIO.output(electrovalve1, GPIO.HIGH)
          GPIO.output(electrovalve2, GPIO.LOW)
          GPIO.output(electrovalve3, GPIO.LOW)
          GPIO.output(electrovalve4, GPIO.LOW)
     elif electrovalvula == 2:
          GPIO.output(electrovalve1, GPIO.LOW)
          GPIO.output(electrovalve2, GPIO.HIGH)
          GPIO.output(electrovalve3, GPIO.LOW)
          GPIO.output(electrovalve4, GPIO.LOW)
     elif electrovalvula == 3:
          GPIO.output(electrovalve1, GPIO.LOW)
          GPIO.output(electrovalve2, GPIO.LOW)
          GPIO.output(electrovalve3, GPIO.HIGH)
          GPIO.output(electrovalve4, GPIO.LOW)
     elif electrovalvula == 4:
          GPIO.output(electrovalve1, GPIO.LOW)
          GPIO.output(electrovalve2, GPIO.LOW)
          GPIO.output(electrovalve3, GPIO.LOW)
          GPIO.output(electrovalve4, GPIO.HIGH)
```

Dicha función recibe como parámetro la variable electroválvula, en función del valor de esta pondrá a 1 la salida correspondiente a la electroválvula que se dese abrir y a 0 el resto. Tal y como se explicó en la Tabla 7.

5.3.1.2. Función arranque del motor

La función recibe como parámetro el valor de la potencia de succión desea, este valor debe de estar entre 50 y 100 como se indicó en la sección [5.2.5]. Ya que, si el valor es menor que 50 el motor no arrancar por no recibir la potencia suficiente.

```
# Arranque motor.

def motor_start(succion):
      PWM.start(motorPin,succion)
```

Tal y como se ha implementado el control del motor de succión, es posible realizar en trabajos futuros modulaciones por flujo de succión. En este proyecto no se implementa, pero se ha tenido en cuenta a la hora de desarrolla la plataforma, y se ha introducido esta posibilidad.

5.3.1.3. Función lectura de temperatura y humedad

La función realiza la lectura de la temperatura y humedad por medio del sensor DHT22, para ello se hace uso de la librería de Adafruit realizada para este sensor.

```
#     Funcion de medida de temperatura y humedad

def measure_tyh(tiempo):

      j=0
      time.sleep(9)
      #Lectura humedad y temperatura
      while (tiempo == True):
            tick_HT = time.time()
            humidity, temperature =
Adafruit_DHT.read_retry(sensorTemp22, Temp22)
            instante = datetime.now()
            tack_HT=time.time()
            t_HT=tack_HT-tick_HT


            #Cuanto duerme en funcion de lo que tarde en H y T
            if t_HT > SLEEP_tyh:
                  print "Tiempo medicion H y T > SLEEP:", t_HT
            else:
                  print "Tiempo medicion H y T:", t_HT


                  if humidity is not None  and temperature is not
None:
                        print'\nSensor DHT22: ' +str(sensorTemp22)
                        print'>>> '+str(instante)+'     Temp = '
+str(temperature)+ '  Humidity = ' +str(humidity)+'\n'
                        h = open(ruta_fichero_tyh, "a")
```

```
                    wline_h=str(instante)+'
    '+str(temperature)+'  '+str(humidity)+'\n'
                    h.writelines(wline_h)
                    h.flush()

               else :
                    print 'Failed to get reading, Try again!'
                    h = open(ruta_fichero_tyh, "a")
                    wline_h=str(instante)+'
    '+str(temperature)+'  '+str(humidity)+'\n'
                    h.writelines(wline_h)
                    h.flush()

          tack=time.time()
          time.sleep(SLEEP_tyh-(tack-tick_HT))
          j+=1
```

Esta función se ejecuta siempre como un hilo secundario, de forma que permite ser introducido y ejecutado en cualquier experimento sin interferir en el programa principal. Para que la ejecución del hilo se realice correctamente es necesario indicar en el script la librería en la que se encuentra el método thread.

```
import thread
```

La llamada a la función no se realiza directamente, sino que se utiliza el método thread, tal y como se muestra a continuación:

```
#Llamada al thread de medida de temperatura y humedad
thread.start_new_thread(measure_tyh, (True,))
```

Para matar el proceso basta con:

```
thread.exit()
```

El código se ha realizado para que se tome una lectura cada minuto y se guarde en un fichero de texto, dado que la temperatura y la humedad son variables del entorno que no cambiando muy bruscamente, pero que nos interesa tener monitorizadas porque influyen en la respuesta del sensor como se indicó en la sección [3.3.1.2].

5.3.2. Adquisición pura sin modulación

Se ha desarrollado un código para experimentaciones sin modulación. El código es síncrono y conmuta las electroválvulas de odorante cada *switch* segundos, dato introducido por parámetro, y realiza una lectura por el puerto ADC cada segundo. Dicho código se puede

encontrar en el Anexo [H.1], aunque en esta sección se detallarán los puntos más importantes del mismo.

La sentencia de ejecución para el experimento es:

```
tgs2600puro.py succion switch tiempo
```

Donde succión, switch y tiempo son tres variables que se introducen por parámetro. El valor de succión es un entero entre 50 y 100, y representa la potencia de succión del motor. Switch es también un entero, que representa el tiempo en segundos que permanece abierta cada electroválvula. Por último, tiempo, es otro entero, que representa la duración del experimento en minutos. En función del valor de tiempo y switch, se establecerá el número de conmutaciones entre válvulas que se realizan.

Con la ejecución del script se lanzan dos procesos, el principal y el proceso de lectura de temperatura y humedad (véase sección [5.3.1.3]).

5.3.2.1. Variables y puertos experimentación pura sin modulación

Esta modulación además de los puertos globales especificados utiliza:

PUERTO	NOMBRE	COMANDO	COMENTARIO
P9_38	sensorPIn2600	`sensorPin2600 = 'P9_38'`	Sensor TGS2600

Tabla 10: *Puertos experimentación pura sin modulación de la plataforma.*

Tanto estos puertos como los que controlan las electroválvulas necesitan configurarse como entrada o salida (esto no pasa con los puertos ADC, que por defecto ya son de entrada).

```
##    DECLARACION DE ENTRADAS/SALIDAS DEL SISTEMA.

GPIO.setup("P8_10",GPIO.OUT)
GPIO.setup("P8_12",GPIO.OUT)
GPIO.setup("P8_14",GPIO.OUT)
GPIO.setup("P8_16",GPIO.OUT)
```

Con las anteriores líneas de código se declaran los cuatro puertos que controlan las electroválvulas como salida. Por lo tanto, tendremos estas 4 salidas más la salida del puerto de medida del sensor.

Además de las variables globales son necesarias variables específicas para esta modulación:

VARIABLES	VALOR	COMANDO	COMENTARIO
SAMPLESINICIO	10	SAMPLESINICIO=2	Capturas iniciales para el calentamiento del sensor
NM	10	NM=10	Número de lecturas ADC
T	0.1	T=0.1	Tiempo en que se hacen las NM lecturas
SLEEP	1	SLEEP=1	Periodo de estabilización cuando se pone una nueva temperatura

Tabla 11: *Variables experimentación pura sin modulación de la plataforma.*

5.3.2.2. Ficheros de salida modulación en amplitud

Los datos capturados, además de guardar en las variables, se almacenan en ficheros. Cada vez que se ejecuta esta modulación se generan 3 ficheros. Dos de ellos sólo de datos y otro con datos y parámetros de la experimentación. Los ficheros se guardan todos bajo la misma ruta: */root/PURO/mes/dia*. Donde mes y día corresponde a la fecha de ejecución del experimento. En la siguiente tabla se muestra el nombre de cada fichero y lo que almacena:

NOMBRE DEL FICHERO		FUNCIÓN
Fecha y hora en formato (%a%d%b%Y-%HH%MM%SS) +	_puroTGS2600.txt	Fichero de información con parámetros y lecturas de la experimentación
	_puroTGS2600.dat	Fichero de salida de datos, almacena las lecturas.
	_TyH_TGS2600.dat	Fichero de salida de datos del sensor de temperatura y humedad

Tabla 12: *Ficheros experimentación pura sin modulación.*

5.3.2.3. Funciones de lectura

El programa lee la variación de resistencia del sensor haciendo uso del puerto ADC de la Beaglebone. Se realiza una captura cada segundo y se conmutan las electroválvulas cada *switch* segundos, por lo que para cada pluma de odorante se capturarán switch muestras. El código de lectura se divide en dos funciones, una inicial que se encarga de capturar un número de muestras indicado por las variables *SAMPLESINICIO*, Estas, que se realizan sin presencia de odorante, sirven para la calentar el sensor y para limpiar el circuito de adquisición. Dado que no se conmutan las electroválvulas para dejar pasar el odorante durante la ejecución de esta función.

Las muestras de esta función también se almacenan en los ficheros por lo que para trabajos futuros se tendrá que tener en cuenta que las *SAMPLESINICIO* muestra iniciales se deben descartar. Estas muestras son fáciles de distinguir en el fichero de información (*puroTGS2600.txt*) puesto que cada línea de captura comienza como: *Muestra_ini[]*.

La función inicial se llama samplesinicio*_puroTGS2600()* está compuesta por un bucle *for* que realiza diez lecturas consecutivas, espaciadas *tsub* segundos. El valor de la captura será la media de estas. Cabe destacar que antes del bucle se hace una primera captura que se descarta por ser errónea debido al bug. Cada captura se almacena con su instante de captura en los ficheros de salida *_puroTGS2600()*.txt y *_puroTGS2600().dat*.

```
ADC.read_raw(sensorPin2600) #la primera medida es erronea por el
bug
for i in range(NM):
     value += ADC.read_raw(sensorPin2600)
     r=random.uniform(0, tsub)
     time.sleep(r)
     residuo += (tsub-r)
time.sleep(residuo);
valueTGS2600=value/NM
instante_captura=datetime.now()
```

5.3.2.4. Función de cierre

Se trata de la última función que se ejecuta en el script. Sirve para verificar se cierren los ficheros, obtener la hora de fin del experimento, matar procesos secundarios y terminar las comunicaciones de los puertos de entrada/salida.

5.3.2.5. Función main

Esta función es la encargada de realizar las llamadas al resto de funciones, comprobar que los parámetros introducidos son correctos.

En primer lugar, comprueba si los datos introducidos por parámetros son válidos, si no es así muestra un mensaje por pantalla e interrumpe la ejecución del script.

```
if(len(sys.argv)<4):
          print '\n\nPARAMETROS INCORRECTOS. \n' \
          'Los parametros deben ser:\n' \
          'Succion motor (50-100) \n' \
          'Tiempo conmutacion electrovalvula (en segundos) \n' \
          'Tiempo de experimentacion (en minutos)\n'
          return 0

     print '\nSensor: TGS2600    Pin ADC: ' +sensorPin2600
     print 'Algoritmo: PURO SIN MODULACION'
     print 'Ruta Fichero: ' +ruta_fichero
     print 'Ruta Fichero de datos: '+ruta_fichero_data
```

Si la sentencia de ejecución es buena, se calcula una transición aleatoria para conmutar las electroválvulas. Se hace uso del método *numpy.random.randint* y se crea un vector aleatorio de apertura de electroválvulas.

```
#Se calcula de forma aleatoria la conmutacion de electrovalvulas
vec_open_valve = numpy.random.randint(1,4,SAMPLES)
```

Donde el primer argumento es el valor de menor índice, el segundo argumento es el valor de máximo índice (no incluido, es decir, el 4 no aparece en el vector) y el tercer argumento es la longitud del vector. Por ejemplo, un vector posible para un valor de *SAMPLES = 5* sería:

vec_open_valve = [1 3 2 3 2]

Se ejecuta función de lectura inicial durante *SAMPLESINICIO* veces, sin presencia de odorante y con la temperatura de calentamiento introducida por parámetro. A continuación, se ejecuta la función de lectura principal (*puro_TGS2600()*) un total de 60 veces el valor introducido por parámetro tiempo, es decir, si se introdujo tiempo igual a 5, se ejecutará 300 veces. Se realiza una conmutación entre electroválvulas cada *switch* muestras o segundos, por lo que se realizaran un total de conmutaciones dado por el siguiente calculo *(tiempo*60)/switch*.

Por ejemplo, si el tiempo introducido es 5 minutos y el switch introducido es de 45 segundos, se tendrán 5*60/45=6,667 conmutaciones. Como se puede observar el valor obtenido no es un entero, el código descartara la parte decimal y realizara 6 conmutaciones.

Por último, al terminar la adquisición, se llama a la función *cierre()* y termina el experimento.

5.3.3. Martinelli. Modulación en frecuencia

Para el desarrollo software de la modulación en frecuencia o Martinelli, se ha realizado un código asíncrono. Dado que la salida del 555 genera una señal cuadrada se ha hecho uso de la librería de Adafruit para detectar a través del GPIO cuando se produce un flanco de subida o bajada. El código se puede ver en el Anexo [H.3], aquí se detallarán los puntos más importantes del mismo.

La sentencia de ejecución para el experimento es:

```
tgs2600martinelli.py succion muestras switch
```

Donde *succión* y *muestras* son dos variables que se introducen por parámetro. El valor de *succión* es un entero entre 50 y 100, y representa la potencia de succión del motor. Mientras que muestras, que también es un entero, se refiere a la cantidad de medidas a realizar. En este punto cabe especificar que tomamos como muestra todos los bloques de *K*

pulsos que entre en switch segundos. Como se indicó en la sección 3.5, se ha fijado el valor de *K* en 16 pulsos. El parámetro *switch* se refiere al tiempo que permanece abierta cada electroválvula, o tiempo de conmutación entre electroválvulas. Al tratarse de un código asíncrono la conmutación entre electroválvulas no se realizará cada *switch* segundos, sino que se efectuará cuando se supere dicho valor.

Por ejemplo, imaginemos que introducimos como tiempo de conmutación entre electroválvulas de 60 segundos y la duración de los primeros *K* pulsos nos da 28. El código continuara con la captura de los siguientes *K* pulsos dejando abierta la misma electroválvula. Si la duración de este segundo bloque de pulsos es mayor de 32 segundos se conmutará a la siguiente electroválvula y se procederá a la captura de una nueva muestra.

Con la ejecución del script se lanzan dos procesos, el principal y el proceso de lectura de temperatura y humedad (véase sección [5.3.1.3]).

5.3.3.1. Variables y puertos modulación en frecuencia

Esta modulación además de los puertos globales especificados utiliza:

PUERTO	NOMBRE	COMANDO	COMENTARIO
P9_12	sensorPIn555	`sensorPin555 = 'P9_12'`	Sensor TGS2600 tras conversión con 555
P9_14	heatPin2600	`heatPin2600 = 'P9_14'`	calentamiento del sensor TGS2600

Tabla 13: *Puertos modulación en frecuencia de la plataforma.*

Tanto estos puertos como los que controlan las electroválvulas necesitan configurarse como entrada o salida (esto no pasa con los puertos ADC y PWM, que por defecto ya son de entrada y salida, respectivamente).

```
##    DECLARACION DE ENTRADAS/SALIDAS DEL SISTEMA.

GPIO.setup("P8_10",GPIO.OUT)
GPIO.setup("P8_12",GPIO.OUT)
GPIO.setup("P8_14",GPIO.OUT)
GPIO.setup("P8_16",GPIO.OUT)
GPIO.setup("P9_12",GPIO.IN)
```

Con las anteriores líneas de código se declaran los cuatro puertos que controlan las electroválvulas como salida y el puerto conectado a la salida del 555 como entrada. Por lo tanto, tenemos 5 salidas y 1 entrada.

Además de las variables globales son necesarias variables específicas para esta modulación:

VARIABLES	VALOR	COMANDO	COMENTARIO
SAMPLESINICIO	1	SAMPLESINICIO=1	Capturas iniciales para el calentamiento del sensor
Up		up=[]	Almacena el tiempo de los pulsos en estado alto
Down		down=[]	Almacena el tiempo de los pulsos en estado bajo
duracion		duracion=[]	Almacena la duración de cada pulso
duracion_k		duracion_k=[]	Almacena la duración de k pulsos

Tabla 14: *Variables modulación en frecuencia de la plataforma.*

5.3.3.2. Ficheros de salida modulación Martinelli

Los datos capturados, además de guardar en las variables, se almacenan en ficheros. Cada vez que se ejecuta esta modulación se generan 4 ficheros. Tres de ellos sólo de datos y otro con datos y parámetros de la experimentación. Los ficheros se guardan todos bajo la misma ruta: */root/FRECUENCIA/MARTINELLI/mes/dia*. Donde mes y día corresponde a la fecha de ejecución del experimento. En la siguiente tabla se muestra el nombre de cada fichero y lo que almacena:

NOMBRE DEL FICHERO		FUNCIÓN
Fecha y hora en formato (%a%d%b%Y-%HH%MM%SS) +	_Martinelli_TGS2600.txt	Fichero de información con parámetros y lecturas de la experimentación
	_Martinelli_data1.dat	Fichero de salida de datos, almacena la duración de los pulsos
	_Martinelli_data2.dat	Fichero de salida de datos, almacena la duración de los k pulsos
	_TyH_TGS2600.dat	Fichero de salida de datos del sensor de temperatura y humedad

Tabla 15: *Ficheros de salida de datos modulación en frecuencia.*

5.3.3.3. Funciones de lectura

El programa detecta la variación de los pulsos de salida del 555, haciendo uso del GPIO. Para la captura de las muestras de odorante se han realizado dos funciones, la primera que se encarga de capturar un número de muestras iniciales indicado por la variable *SAMPLESINCIO*. Estas capturas se realizan sin presencia de odorante y sirven para que el sensor se caliente y para limpiar el circuito de adquisición. Dado que no se conmutan las electroválvulas para dejar pasar el odorante durante la ejecución de esta función. Las muestras de esta función también se almacenan en los ficheros por lo que para trabajos futuros se tendrá que tener en cuenta que las *SAMPLESINICIO* muestra iniciales se deben

descartar. Estas muestras son fáciles de distinguir en el fichero de información *(Martinelli_TGS2600.txt)* puesto que cada línea de captura comienza como: *Muestra_ini[]*.

La función inicial se llama *samplesinicio_martinelli_TGS2600()* está compuesta con dos bucles while. El primero se ejecuta durante 8 pulsos y fija la temperatura de calentamiento a 60. El segundo recoge los siguientes 8 pulsos y fijo la temperatura de calentamiento a 100.

Para detectar un cambio de pulso en la salida del 555 se utiliza el método *wait_for_edge*, a continuación, se muestra un ejemplo:

```
#Se espera un pulso de bajada
value_down = GPIO.wait_for_edge(sensorPin555, GPIO.FALLING)

#Se espera un pulso de subida
value_up = GPIO.wait_for_edge(sensorPin555, GPIO.RISING)
```

El método hace que el GPIO indicado se quede en espera de la llegada de un pulso de bajada (FALLING) o de subida (RISING), hasta que eso no sucede la ejecución se interrumpe. Cada vez que se produce una transición se toma el instante temporal en el que se produjo, para poder calcular los tiempos en estado alto y bajo de la señal de salida del 555.

La segunda función se encarga de la lectura de las muestras de odorante de la experimentación y se llama *martinelli_TGS2600()*. Es igual que la función inicial, pero mientras esta se ejecuta las electroválvulas ya permiten el paso de odorante. El total de muestras de odorante serán introducidas por parámetro y se guardan en las variables *SAMPLES*.

5.3.3.4. Función de cierre

Se trata de la última función que se ejecuta en el script. Sirve para verificar se cierren los ficheros, obtener la hora de fin del experimento, matar procesos secundarios y terminar las comunicaciones de los puertos de entrada/salida.

5.3.3.5. Función main

Esta función es la encargada de realizar las llamadas al resto de funciones, comprobar que los parámetros introducidos son correctos y almacenar los datos de captura en el fichero de datos *Martinelli_data2.dat* y el fichero de información *Martinelli_TGS2600.txt*.

En primer lugar, comprueba si los datos introducidos por parámetros son válidos, si no es así muestra un mensaje por pantalla e interrumpe la ejecución del script.

```
if(len(sys.argv)<3):
          print '\n\nPARAMETROS INCORRECTOS. \n' \
          'Los parametros deben ser:\n' \
```

```
            'Succion motor (50-100) \n' \
            'Duracion de la experimentacion (en muestras)\n'\
            'Tiempo de apertura de cada válvula (en segundos)'
        return 0

        print '\nSensor: TGS2600    Pin ADC: ' +sensorPin2600
        print 'Algoritmo: MARTINELLI'
        print 'Ruta Fichero: ' +ruta_fichero
        print 'Ruta Fichero de datos 1: '+ruta_fichero_data1
        print 'Ruta Fichero de datos 2: '+ruta_fichero_data2
```

Si la sentencia de ejecución es buena, se calcula una transición aleatoria para conmutar las electroválvulas. Se hace uso del método numpy.random.randint y se crea un vector aleatorio de apertura de electroválvulas. Cuyo código y funcionamiento es el mismo que el explicado en la sección [5.3.2.5].

Se ejecuta la función inicial de captura *SAMPLESINICIO* veces consecutivas, y en cada iteración se almacena el valor de la duración de los k=16 pulsos en la variable *duración_k*. A continuación, se ejecuta la función de lectura principal (*martinelli_TGS2600()*), esta será llamada *SAMPLES* veces y en cada iteración se producirá un cambio de electroválvula, si el tiempo *switch* se ha superado, de acuerdo con el vector aleatorio de apertura de electroválvulas. Esta función también almacena, después de cada iteración, en la variable *duración_k* el valor de los 16 pulsos.

Por último, al terminar la adquisición, se llama a la función *cierre()* y termina el experimento.

5.3.4. Regresión temperatura

El desarrollo software de la modulación por amplitud (regresión temperatura) se ha realizado en Python. Se ha elaborado un código síncrono que conmuta las electroválvulas de odorante cada minuto y realiza una lectura de odorante por el puerto ADC cada segundo. El código se puede ver en el Anexo [H.3], en esta sección se detallarán los puntos más importantes del mismo.

La sentencia de ejecución para el experimento es:

```
tgs2600regresion.py succion temperatura tiempo switch
```

Donde succión, temperatura y muestras son tres variables que se introducen por parámetro. El valor de succión es un entero entre 50 y 100, y representa la potencia de succión del motor. Temperatura es también un entero, pero en este caso entre 0 y 100, que representa la temperatura de calentamiento del sensor. Por último, tiempo, es otro entero, que representa la duración del experimento en minutos.

Con la ejecución del script se lanzan dos procesos, el principal y el proceso de lectura de temperatura y humedad (véase sección [5.3.1.3]).

5.3.4.1. Variables y puertos modulación en amplitud

Esta modulación además de los puertos globales especificados utiliza:

PUERTO	NOMBRE	COMANDO	COMENTARIO
P9_40	sensorPin2600	`sensorPin2600 = 'P9_40'`	Sensor TGS2600
P9_22	heatPin2600	`heatPin2600 = 'P9_22'`	Calentamiento del sensor TGS2600

Tabla 16: *Puertos modulación en amplitud de la plataforma.*

En este caso los únicos puertos que necesitan configurarse como entradas o salidas son los que controlan las electroválvulas. El resto no será necesario especificarlos como entrada o salida, ya que se trata de un puerto PWM (siempre salida) y un puerto ADC (siempre entrada). El puerto ADC sin embargo si necesita un setup previo a su uso:

```
ADC.setup()
```

Además de las variables globales son necesarias variables específicas para esta modulación:

VARIABLES	VALOR	COMANDO	COMENTARIO
SAMPLESINICIO	10	`SAMPLESINICIO=10`	Capturas iniciales para el calentamiento del sensor
NM	10	`NM=10`	Número de lecturas ADC
T	0.1	`T=0.1`	Tiempo en que se hacen las NM lecturas
SLEEP	1	`SLEEP=1`	Periodo de estabilización cuando se pone una nueva temperatura
TENDENCIA	5	`TENDENCIA=5`	Tendencia de la modulación de temperatura

Tabla 17: *Variables modulación en amplitud de la plataforma.*

5.3.4.2. Ficheros de salida modulación en amplitud

Los datos capturados, además de guardar en las variables, se almacenan en ficheros. Cada vez que se ejecuta esta modulación se generan 3 ficheros. Dos de ellos sólo de datos y otro con datos y parámetros de la experimentación. Los ficheros se guardan todos bajo la misma ruta: */root/AMPLITUD/REGRESIÓNmes/dia*. Donde mes y día corresponde a la fecha de ejecución del experimento. En la siguiente tabla se muestra el nombre de cada fichero y lo que almacena:

NOMBRE DEL FICHERO		FUNCIÓN
Fecha y hora en formato (%a%d%b%Y-%HH%MM%SS) +	_Regresion_TGS2600.txt	Fichero de información con parámetros y lecturas de la experimentación
	_Regresion_TGS2600.dat	Fichero de salida de datos, almacena las lecturas.
	_TyH_TGS2600.dat	Fichero de salida de datos del sensor de temperatura y humedad

Tabla 18: *Ficheros de salida de datos modulación en amplitud.*

5.3.4.3. Funciones de lectura

El programa lee la variación de resistencia del sensor haciendo uso del puerto ADC de la Beaglebone. Se realiza una captura cada segundo y se conmutan las electroválvulas cada *switch* segundos, por lo que para cada pluma de odorante se capturarán *switch* muestras. El código de lectura se divide en dos funciones, una inicial que se encarga de capturar un número de muestras indicado por las variables *SAMPLESINICIO*, Estas, que se realizan sin presencia de odorante, sirven para la calentar el sensor y para limpiar el circuito de adquisición. Dado que no se conmutan las electroválvulas para dejar pasar el odorante durante la ejecución de esta función.

Las muestras de esta función también se almacenan en los ficheros por lo que para trabajos futuros se tendrá que tener en cuenta que las *SAMPLESINICIO* muestra iniciales se deben descartar. Estas muestras son fáciles de distinguir en el fichero de información (*Regresión_TGS2600.txt*) puesto que cada línea de captura comienza como: *Muestra_ini[]*.

La función inicial se llama *samplesinicio_regresion_TGS2600()* está compuesta por un bucle for que realiza diez lecturas consecutivas, espaciadas *tsub* segundos. El valor de la captura será la media de estas. Cabe destacar que antes del bucle se hace una primera captura que se descarta por ser errónea debido al bug. Cada captura se almacena con su instante de captura en los ficheros de *salida _Regresion_TGS2600().txt* y *_Regresion_TGS2600().dat*.

```
ADC.read_raw(sensorPin2600) #la primera medida es erronea por el
bug
for i in range(NM):
     value += ADC.read_raw(sensorPin2600)
     r=random.uniform(0, tsub)
     time.sleep(r)
     residuo += (tsub-r)
time.sleep(residuo);
valueTGS2600=value/NM
instante_captura=datetime.now()
```

La segunda función de lectura es *regresión_TGS2600(),* que realiza la lectura igual que la primer, y donde se realiza la modulación de la temperatura de calentamiento del sensor de acuerdo a una regresión lineal de las muestras previas de odorante. Para ello se utiliza la función linregress del submódulo SciPy dedicado a la estadística. La función linregress toma como argumentos dos arrays x e y, que toma como dos conjuntos de medidas que deben tener la misma longitud. El array x se denomina igual en el código e indica el número de muestra, el array y es el vector *concentTGS2600*. A continuación, se muestra el fragmento del código:

```
#Adaptacion temperatura
slope, intercept, r_value, p_value, std_err1 =
stats.linregress(x[(count-SAMPLESINICIO):(count-
1)],concentTGS2600[(count-SAMPLESINICIO):(count-1)])
temperature_TGS2600 = heat2600 - (slope*TENDENCIA)

if temperature_TGS2600 < 10.0:
     temperature_TGS2600 = 10.0
elif temperature_TGS2600 >90.0:
     temperature_TGS2600 = 90.0

#Reset de setup PWM
PWM.set_duty_cycle(heatPin2600, temperature_TGS2600)
```

De los resultados que devuelve la función linregress nos interesa el slope, es decir, la pendiente. Esta junto con la variable tendencia se utilizan para calcular la nueva temperatura de calentamiento.

5.3.4.4. Función de cierre

Se trata de la última función que se ejecuta en el script. Sirve para verificar se cierren los ficheros, obtener la hora de fin del experimento, matar procesos secundarios y terminar las comunicaciones de los puertos de entrada/salida.

5.3.4.5. Función main

Esta función es la encargada de realizar las llamadas al resto de funciones, comprobar que los parámetros introducidos son correctos.

En primer lugar, comprueba si los datos introducidos por parámetros son válidos, si no es así muestra un mensaje por pantalla e interrumpe la ejecución del script.

```
if(len(sys.argv)<4):
     print '\n\nPARAMETROS INCORRECTOS. \n' \
     'Los parametros deben ser:\n' \
     'Succion motor (50-100) \n' \
```

```
      'Temperatura Promedio TGS2600 (1-100) \n' \
      'Tiempo de experimentacion (en minutos)\n'
      return 0

print '\nSensor: TGS2600    Pin ADC: ' +sensorPin2600
print 'Algoritmo: REGRESION TEMPERATURA'
print 'Ruta Fichero: ' +ruta_fichero
print 'Ruta Fichero de datos: '+ruta_fichero_data
```

Si la sentencia de ejecución es buena, se calcula una transición aleatoria para conmutar las electroválvulas. Se hace uso del método *numpy.random.randint* y se crea un vector aleatorio de apertura de electroválvulas. Cuyo código y funcionamiento es el mismo que el explicado en la sección [5.3.2.55.3.3.5].

Se ejecuta función de lectura inicial durante *SAMPLESINICIO* veces, sin presencia de odorante y con la temperatura de calentamiento introducida por parámetro. A continuación, se ejecuta la función de lectura principal (*regresion_TGS2600()*) un total de 60 veces el valor introducido por parámetro tiempo, es decir, si se introdujo tiempo igual a 5, se ejecutará 300 veces. Cada ejecución de esta función tarda aproximadamente un *switch* segundos, es decir, se toman lecturas cada segundo.

Por último, al terminar la adquisición, se llama a la función *cierre()* y termina el experimento.

5.3.5. Limpieza

A parte de los dos códigos de modulación, se ha desarrollado otro código para realizar limpieza del circuito de adquisición de odorante. Se ha realizado para eliminar posibles impurezas, residuos de odorante, que puedan quedar tras una experimentación. Es recomendable su ejecución después de realizar un experimento con muestras de odorante, para así asegurar que en la siguiente experimentación el circuito esté limpio.

La sentencia de ejecución del código es:

```
limpieza.py electrovalvula tiempo
```

Tiene dos variables que se introducen por parámetro, electroválvula y tiempo. El valor de la electroválvula puede ser un número del cero al tres. La variable tiempo será la duración en minutos de la limpieza.

- Electroválvula = 0. Indica que todas las válvulas, se conmutarán las válvulas de forma escalada y se limpiarán todos los conductos.
- Electroválvula = 1. Indica que se conmuta la válvula 1 y limpiará el conductor desde la primera válvula hasta el motor de succión.
- Electroválvula = 2. Indica que se conmuta la válvula 1 y limpiará el conductor desde la segunda válvula hasta el motor de succión.

- Electroválvula = 3. Indica que se conmuta la válvula 1 y limpiará el conductor desde la tercera válvula hasta el motor de succión.

El código de limpieza se puede ver en el Anexo [H.4].

6. Resultados y validación

6.1. Introducción

Como se ha venido explicando en temas anteriores se ha desarrollado una plataforma de experimentación para la adquisición de odorante. La plataforma consta de 3 tipos de experimentos diferenciados por su tipo de modulación.

1. Sin modulación.
2. Con modulación en amplitud.
3. Con modulación en frecuencia.

Para cada tipo de experimentación se ha desarrollado una parte hardware y otra parte software, que se han explicado en las secciones [5.2] y [5.3], respectivamente. En la presente sección se va explicar cómo realizar un experimento de cada tipo y se comprobará el correcto funcionamiento de la plataforma para cada tipo de modulación. Se van a probar unos de las funcionalidades más significativas de la plataforma.

Nota: para una fácil distinción de los sensores TGS2600 se han pintado con las letras A, F y W, en función de su circuito de adaptación. Para experimentaciones en amplitud se ha pintado la letra A, para experimentaciones en frecuencia la letra F y para experimentaciones sin modulación la letra W.

6.1.1. Sustancias a detectar. Odorantes

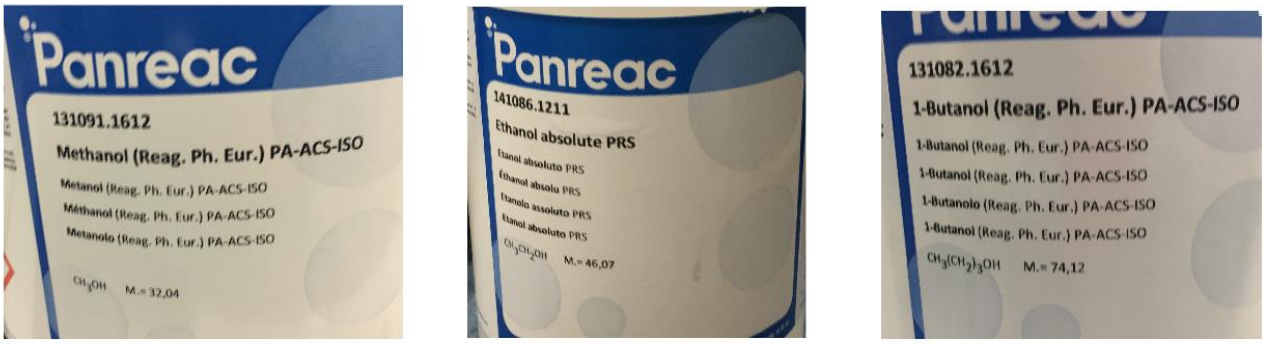
Las sustancias que se han utilizado para la experimentación con la plataforma se muestran en el Figura 6. 1. Se trata de Etanol, Metanol y Butanol. Para la realización de las experimentaciones para la validación de resultados se han introducido en los tubos:

- Tubo 1 (Electroválvula 1): Metanol al 50% de concentración (5 mm de metanol con 5 mm de agua destilada).

- Tubo 2 (Electroválvula 2): Etanol al 50% de concentración (5 mm de etanol con 5 mm de agua destilada).

- Tubo 3 (Electroválvula 3): Butanol al 100% de concentración (10 mm de Butanol)

Se han elegido estos odorantes para realizar las pruebas porque fueron los utilizados en [1][2] y sabemos que los sensores utilizados responden a ellos.



Metanol **Etanol** **Butanol**

Figura 6. 1: *Sustancias utilizadas como odorantes. Metanol, Etanol y Butanol de la empresa Panreac.*

6.2. Adquisición pura sin modulaciones

El circuito realizado en la sección [5.2.1] permite realizar capturas de odorante sin realizar modulaciones en la temperatura de calentamiento del sensor. Será útil para estudiar cómo se comporta el sensor en presencia de un odorante sin actuar sobre su componente de calentamiento y que estos resultados sirvan para comparar con los obtenidos al modular.

6.2.1. Experimento puro sin modulación

A continuación, se va a detallar el procedimiento para lanzar un experimento sin modulación en la plataforma. Para ello se explicarán tanto los pasos a seguir como las comprobaciones que son necesarias realizar. Por ahora la plataforma dispone de un script de captura sin modulación. El funcionamiento de dicho código fue explicado en la sección [5.3.2], y se encuentra tanto en la carpeta raíz de la BBB como en la tarjeta microSD (backup). El procedimiento que se sigue es el siguiente:

1. Encender la BBB y situarse en su directorio raíz (/root). Comprobar que la tarjeta microSD está introducida en su ranura, los ficheros de salida se guardan en ella.

2. Comprobar que todos los conectores están en su sitio, en la siguiente tabla se muestra la relación de conexión. Además, para facilitar la conexión los cables tienen una pegatina con el nombre del pin P8 o P9 al que van conectados.

PUERTO	NOMBRE	COMANDO	COMENTARIO
P8_10	electrovalve1	`electrovalve1 = 'P8_10'`	Válvula 1 (METANOL)
P8_12	electrovalve2	`electrovalve2 = 'P8_12'`	Válvula 2 (ETANOL)
P8_14	electrovalve3	`electrovalve3 = 'P8_12'`	Válvula 3 (BUTANOL)
P8_16	electrovalve4	`electrovalve4 = 'P8_14'`	Válvula 4 (AIRE)
P9_21	motorPin	`motorPin = 'P9_21'`	Motor de succión
P8_11	Temp22	`Temp22 = 'P8_11'`	Sensor de temperatura/humedad
P9_38	sensorPin2600	`sensorPin2600 = 'P9_38'`	Sensor TGS2600
P8_13	heatPin2600	`heatPin2600 = 'P8_13'`	Calentamiento del sensor TGS2600

Tabla 19: *Pines de conexión para experimentación sin modulación*

3. Conectar el transformador de 6V a la plataforma y el de 5V si no están conectados.

4. Situar los botes de odorante en su lugar y conectar el tubo de cada electroválvula al bote adecuado. Dependiendo del experimento a realizar igual se necesitan otros odorantes, por defecto se han utilizado metanol, etanol y butanol.

5. Situar el sensor TGS2600 suministrado para no modulaciones en su posición dentro del circuito de adquisición. Para su rápida distinción se ha pintado una W (*without modulation*) en el sensor.

6. Ejecutar la sentencia de ejecución. Para el caso de la experimentación sin modulación es (véase el Anexo [H.1]):

```
tgs2600puro.py succion switch tiempo
```

   Donde *succión* es la velocidad de absorción del motor que tiene que tener un valor entre 50 y 100. *Switch* es el tiempo en segundos que se mantendrá cada electroválvula abierta y, por último, *tiempo* es la duración de la experimentación.

7. La plataforma realizará la experimentación según lo explicado en la sección [5.3.2], durante el tiempo indicado por parámetro.

8. Tras finalizar la experimentación es conveniente apagar la plataforma si no se va a continuar utilizando. Para su desconexión realizar los siguientes pasos expuesto en el siguiente apartado.

6.2.1.1. Desconexión de la plataforma

1. Apagar la BBB mediante la sentencia: shutdown -h now

2. Esperar hasta que se apague la BBB, todos sus LEDs se apagarán.

3. Desconectar el transformador de 6V.

4. Desconectar el transformador de 5V y la conexión USB o Ethernet, en función de la conexión ssh que se haya realizado.

6.2.1.2. Ejemplo experimento puro sin modulación

Se realizan los pasos indicados en la sección [6.2.1] hasta el punto 12. Como deseamos realizar una experimentación con odorante nos cercioramos que los odorantes estén conectados al circuito. En este caso vamos a realizar una conmutación de odorantes entre Metanol, Etanol y Aire. Aprovechamos la electroválvula 3 para el aire, dejándola libre, sin conectar ningún odorante.

Continuamos con paso 13 y se ejecuta la siguiente sentencia que inicia la experimentación.

```
tgs2600puro.py 100 30 2
```

Para la realización del experimento se ejecuta el script puro con valor de velocidad de succión 100, es decir, succión máxima, tiempo de conmutación entre electroválvulas 30 segundos y se capturaran durante 2 minutos. En la gráfica de la Figura 6. 2 se puede observar la salida obtenida, se representa los 120 + SAMPLESINICIO segundos del experimento. Se conmutan las electroválvulas según su vector de conmutación entre válvulas: Conmutación entre electroválvulas: [1 3 2 1]. Donde los números 1, 2 y 3 son las electroválvulas (1-METANOL, 2-ETANOL, 3-BUTANOL, 4-AIRE).

Cada 30 segundos se produce la conmutación de electroválvulas, en la gráfica se han señalado los intervalos de apertura de cada válvula. Se aprecia claramente que la entrada de cada odorante hace variar la resistencia interna del sensor y con ello origina un pico de tensión en la señal de salida medida. Se aprecia que los odorantes se distinguen.

A la vista de la Figura 6. 2. Si nos fijamos al producirse el cambio de electroválvula, la señal de salida no aumenta directamente, sino que continúa bajando unos segundos hasta que comienza la rampa de subida. Esto no debería ocurrir, dado que al llegar un nuevo odorante el sensor debería reaccionar disminuyendo su resistencia interna y por lo tanto aumentado el voltaje medido. Esto no quiere decir que este mal sistema, sino que la demora en la detección del odorante puede ser fruto del aire que contienen los tubos en su interior. Sobre todo, en las primeras medidas, puedes el aire de los tubos tendrá menos densidad de odorante. Se ha incluido esta gráfica en la memoria para evidenciar que las primeras medidas de los experimentos pueden ser erróneas.

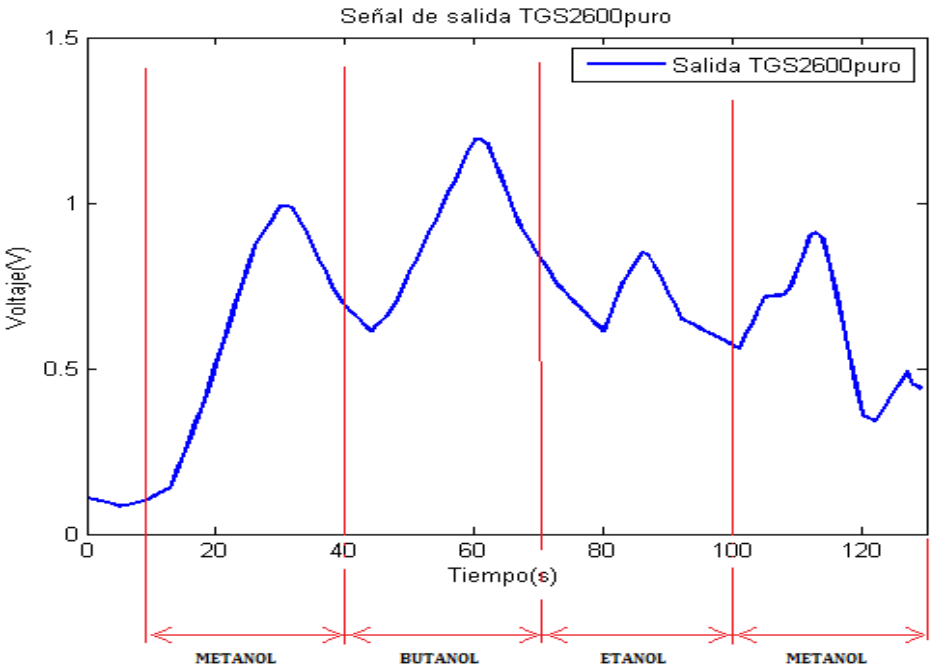


Figura 6. 2: *Datos obtenidos de la experimentación pura sin modulación.*

Para comprobar si es debido a esto, se ha ejecutado otra experimentación, mucho más larga con transiciones continuas, con 60 segundos de apertura de válvula, entre Metanol, Etanol y Butanol. En la Figura 6. 3 y Figura 6. 4 se pueden observar los resultados obtenidos, en ellos se ve que no se produce el fenómeno antes mencionado.

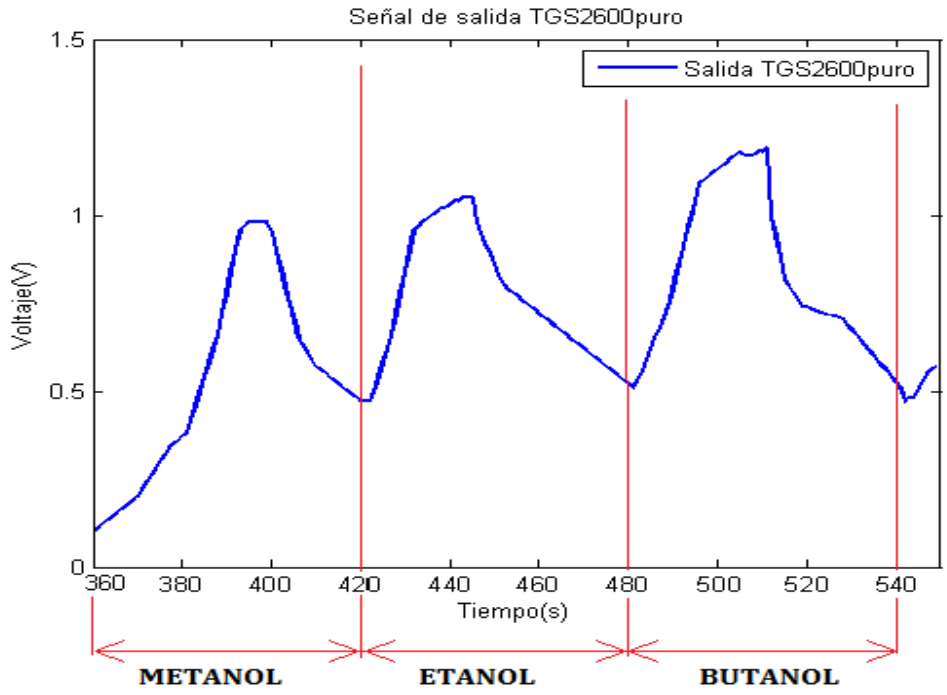


Figura 6. 3: *Gráfica de salida de experimentación pura sin modulación con conmutación de odorantes Metanol, Etanol y Butanol. Tiempo de 360 a 540 segundos.*

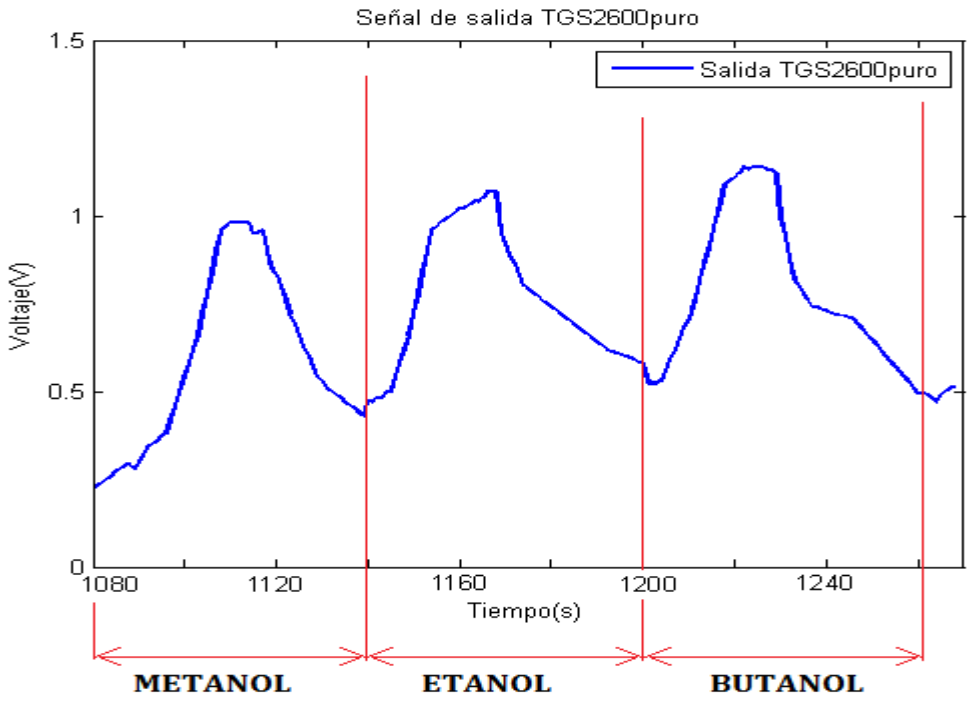


Figura 6. 4: *Gráfica de salida de experimentación pura sin modulación con conmutación de odorantes Metanol, Etanol y Butanol. Tiempo [1080 a 1260 segundos]*

En las gráficas de la Figura 6. 3 y Figura 6. 4 se evidencia con claridad que cada odorante deja una huella bastante característica, que permite diferenciarlos. Se han incluido en la memoria las dos gráficas para demostrar que las huellas son reproducibles.

6.3. Modulación en amplitud

El circuito realizado en la sección [5.2.3] permite realizar modulaciones en amplitud sobre la temperatura de calentamiento del sensor TGS2600. En la sección [5.3.4] se ha explicado un código que modula por regresión en amplitud, este es sólo un ejemplo de una de las posibles modulaciones que se pueden realizar sobre la amplitud. En esta sección se explicará paso por paso como realizar una modulación en amplitud utilizando este código.

Es importante tener claro que la modulación por regresión no es la única modulación en amplitud que se puede realizar, se ha implementado esta para comprobar que la plataforma funciona correctamente y cumple con los objetivos, pero se podrá desarrollar cualquier otro tipo de modulación en amplitud sobre el circuito de la sección [5.2.3] sólo con crear un nuevo script Python para la experimentación.

6.3.1. Experimento de modulación en amplitud por regresión de temperatura

Se va a detallar el procedimiento para lanzar un experimento de modulación en amplitud en la plataforma. Para ello se explicarán tanto los pasos a seguir como las comprobaciones que son necesarias realizar. Además, se ejecutarán dos experimentaciones de adquisición, una sin odorante y otra en presencia de odorante para evidenciar que la plataforma esta prepara para experimentar con ella y realizar otros proyectos de discriminación y la clasificación de odorantes.

Por ahora la plataforma dispone de un script de modulación en amplitud que hace uso de la regresión lineal para adaptar la temperatura de calentamiento del sensor. El funcionamiento de dicho código fue explicado en la sección [5.3.4], y se encuentra tanto en la carpeta raíz de la BBB como en la tarjeta microSD (backup). El procedimiento que se sigue es el siguiente:

1. Encender la BBB y situarse en su directorio raíz (/root). Comprobar que la tarjeta microSD está introducida en su ranura, los ficheros de salida se guardan en ella.

2. Comprobar que todos los conectores están en su sitio, en la siguiente tabla se muestra la relación de conexión. Además, para facilitar la conexión los cables tienen una pegatina con el nombre del pin P8 o P9 al que van conectados.

3. Conectar el transformador de 6V a la plataforma y el de 5V si no están conectados.

PUERTO	NOMBRE	COMANDO	COMENTARIO
P8_10	electrovalve1	`electrovalve1 = 'P8_10'`	Válvula 1 (METANOL)
P8_12	electrovalve2	`electrovalve2 = 'P8_12'`	Válvula 2 (ETANOL)
P8_14	electrovalve3	`electrovalve3 = 'P8_12'`	Válvula 3 (BUTANOL)
P8_16	electrovalve4	`electrovalve4 = 'P8_14'`	Válvula 4 (AIRE)
P9_21	motorPin	`motorPin = 'P9_21'`	Motor de succión
P8_11	Temp22	`Temp22 = 'P8_11'`	Sensor de temperatura/humedad
P9_40	sensorPin2600	`sensorPin2600 = 'P9_40'`	Sensor TGS2600
P9_22	heatPin2600	`heatPin2600 = 'P9_22'`	Calentamiento del sensor TGS2600

Cuadro 1: *Pines de conexión para experimentación de modulación en amplitud por regresión de temperatura*

4. Situar los botes de odorante en su lugar y conectar el tubo de cada electroválvula al bote adecuado. Dependiendo del experimento a realizar igual se necesitan otros odorantes, por defecto se han utilizado metanol, etanol y butanol.

5. Situar el sensor de modulaciones en amplitud en su posición dentro del circuito de adquisición. Para su rápida distinción se ha pintado una A en el sensor.

6. Ejecutar la sentencia de ejecución. Para el caso de la modulación en amplitud por regresión es (véase el Anexo [H.3]):

```
tgs2600regresion.py succion temperatura tiempo switch
```

   Donde *succión* es la velocidad de absorción del motor que tiene que tener un valor entre 50 y 100. *Temperatura* es el valor para la temperatura de calentamiento del sensor, valor en porcentaje entre 0 y 100. *Switch* es el tiempo en segundos que se mantendrá cada electroválvula abierta y, por último, *tiempo* es la duración de la experimentación.

7. La plataforma realizará la experimentación según lo explicado en la sección [5.3.4], durante el tiempo indicado por parámetro.

8. Tras finalizar la experimentación es conveniente apagar la plataforma si no se va a continuar utilizando. Para su desconexión realizar los siguientes pasos expuesto en el siguiente apartado.

6.3.1.1. Desconexión de la plataforma

1. Apagar la BBB mediante la sentencia: shutdown -h now

2. Esperar hasta que se apague la BBB, todos sus LEDs se apagarán.

3. Desconectar el transformador de 6V.

4. Desconectar el transformador de 5V y la conexión USB o Ethernet, en función de la conexión ssh que se haya realizado.

6.3.1.2. Ejemplo experimento de regresión de temperatura con odorante

Se realizan los pasos indicados en la sección [6.3.1] hasta el punto 12. Como deseamos realizar una experimentación con odorante nos cercioramos que los odorantes estén conectados al circuito. En este caso vamos a realizar una conmutación de odorantes entre Metanol, Etanol y Aire. Aprovechamos la electroválvula 3 para el aire, dejándola libre, sin conectar ningún odorante.

Continuamos con paso 13 y se ejecuta la siguiente sentencia que inicia la experimentación.

```
tgs2600regresion.py 100 90 2 30
```

Para la realización del experimento se introduce una velocidad de succión del 100 por cien, es decir, succión máxima, y un porcentaje de temperatura de 90. Se fija el tiempo de

experimentación en 2 minutos y el tiempo de conmutación entre electroválvulas en 30 segundos. El experimento durará 130 segundos, los dos minutos introducidos más las 10 muestras iniciales. Esto quiere decir que se realizarán 4 conmutaciones de válvulas. En la Figura 6. 5 se representa la salida obtenida en función de la temperatura de calentamiento en cada instante de tiempo.

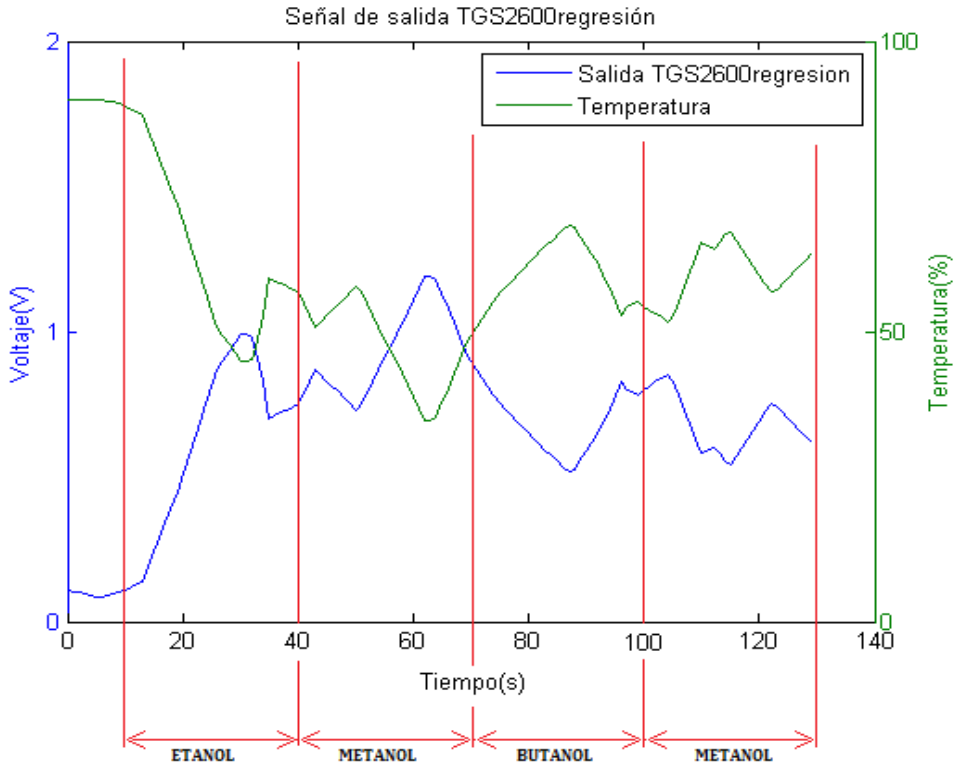


Figura 6. 5: *Salida experimentación Regresión temperatura*.

La conmutación entre electroválvulas ha sido [2 1 3 1]. Donde los números 1, 2 y 3 son las electroválvulas (1-METANOL, 2-ETANOL, 3-BUTANOL, 4-AIRE). Las primeras 10 muestras corresponden a las *SAMPLESINICIO* muestras de calentamiento. La primera pluma de odorante entra en el segundo 11 y origina un incremento en la señal de voltaje, La temperatura de calentamiento se disminuye con el fin de compensar esta variación de voltaje. Cuando se produce una bajada en la señal de voltaje se realiza el proceso inverso, aumentando la temperatura de calentamiento, para conseguir estabilizar la señal.

Se han realizado otras dos experimentaciones, más largas, con el fin de ver cómo se comporta el sensor cuando ya se ha estabilizado más. En las Figura 6. 6 y Figura 6. 7 se muestra los resultados obtenidos para una transición de Metanol, Etanol y Butanol, cada odorante se aplica durante 60 segundos.

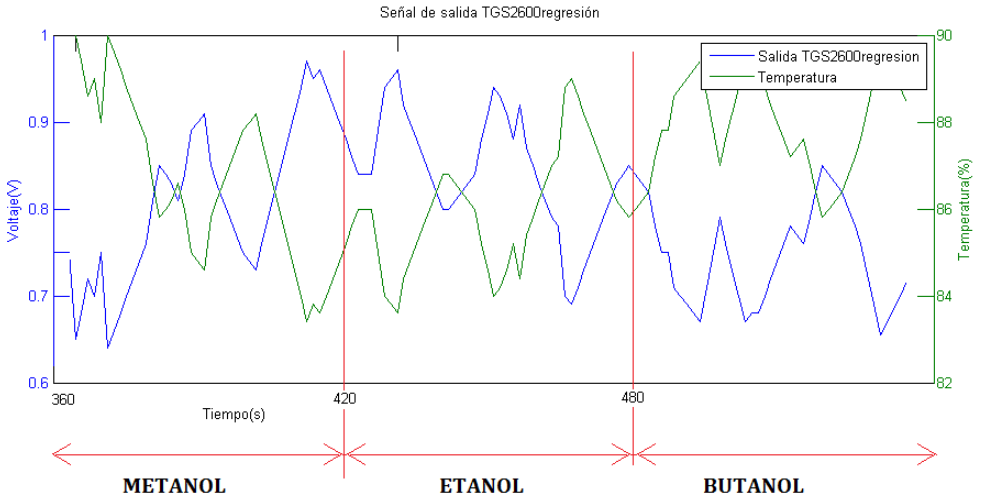


Figura 6. 6: *Gráfica de salida de la experimentación por Regresión de temperatura. Prueba 1 con Metanol, Etanol y Butanol.*

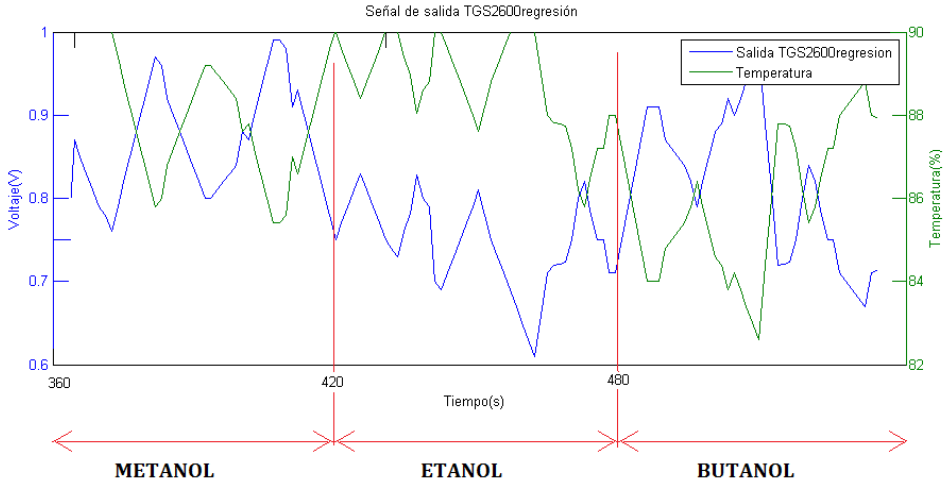


Figura 6. 7: *Gráfica de salida de la experimentación por Regresión de temperatura. Prueba 2 con Metanol, Etanol y Butanol.*

A la vista de las Figura 6. 7 y Figura 6. 8, no es nada intuitiva la detección del odorante. Esto se debe a que será necesario ajustar el parámetro *TENDENCIA*. En estea experimentación lo fijamos en 5, por ello la señal varia tanto. Para ajustar el parámetro se tendrán que realizar diferentes y largas experimentaciones para poder comparar y realizar un ajuste óptimo. Aunque no era el objetivo del proyecto, se han realizado varias experimentaciones con este fin y los mejores resultados obtenido se muestran en la siguiente gráfica. Donde se puede apreciar que la señal varia menos con el paso del tiempo, empieza a estabilizarse.

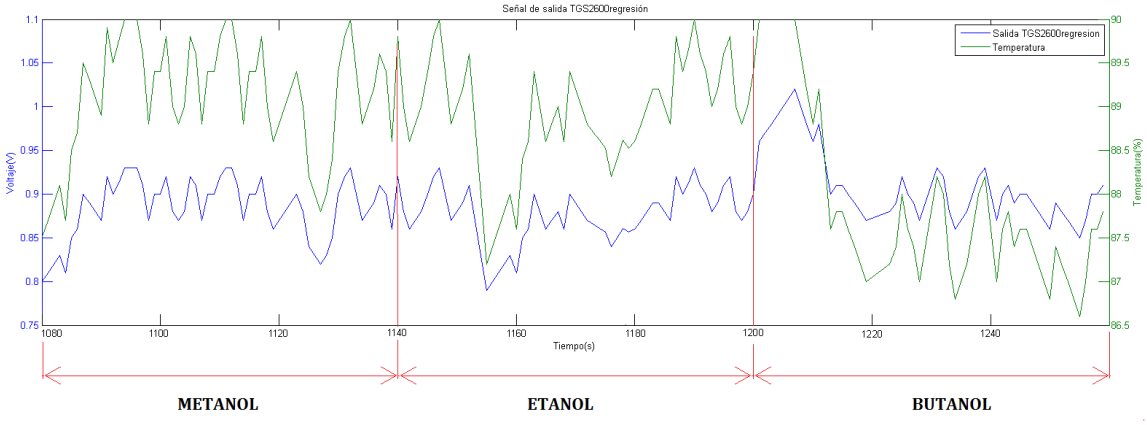


Figura 6. 8: Gráfica de salida experimentación de regresión. Representadas tres conmutaciones de electroválvulas.

6.4. Modulación en frecuencia

El circuito realizado en la sección [5.2.2] permite realizar modulaciones en frecuencia sobre la temperatura de calentamiento del sensor TGS2600. En la sección [5.3.3] se ha explicado un código que modula de esta manera, denominado Martinelli, este es sólo un ejemplo de una de las posibles modulaciones que se pueden realizar sobre la frecuencia. En esta sección se explicará paso por paso como realizar una modulación en frecuencia utilizando este código.

Es importante tener claro que la modulación por Martinelli no es la única modulación en frecuencia que se puede realizar, se ha implementado esta para comprobar que la plataforma funciona correctamente y cumple con los objetivos, pero se podrá desarrollar cualquier otro tipo de modulación en frecuencia sobre el circuito de la sección [5.2.2][5.2.3] sólo con crear un nuevo script Python para la experimentación.

6.4.1. Experimento de modulación en frecuencia por método Martinelli

Se va a detallar el procedimiento para lanzar un experimento de modulación en frecuencia en la plataforma. Para ello se explicarán tanto los pasos a seguir como las comprobaciones que son necesarias realizar. Además, se ejecutarán dos experimentaciones de adquisición, una sin odorante y otra en presencia de odorante para evidenciar que la plataforma esta prepara para experimentar con ella y realizar otros proyectos de discriminación y la clasificación de odorantes.

Por ahora la plataforma dispone de un script de modulación en frecuencia que fija el valor de la temperatura de calentamiento del sensor al sesenta o ciento por ciento en función

de la frecuencia de la señal de salida. El funcionamiento de dicho código fue explicado en la sección [5.3.3], y se encuentra tanto en la carpeta raíz de la BBB como en la tarjeta microSD (backup). El procedimiento que se sigue es el siguiente:

1. Encender la BBB y situarse en su directorio raíz (/root). Comprobar que la tarjeta microSD está introducida en su ranura, los ficheros de salida se guardan en ella.

2. Comprobar que todos los conectores están en su sitio, en la siguiente tabla se muestra la relación de conexión. Además, para facilitar la conexión los cables tienen una pegatina con el nombre del pin P8 o P9 al que van conectados.

PUERTO	NOMBRE	COMANDO	COMENTARIO
P8_10	electrovalve1	`electrovalve1 = 'P8_10'`	Válvula 1 (METANOL)
P8_12	electrovalve2	`electrovalve2 = 'P8_12'`	Válvula 2 (ETANOL)
P8_14	electrovalve3	`electrovalve3 = 'P8_12'`	Válvula 3 (BUTANOL)
P8_16	electrovalve4	`electrovalve4 = 'P8_14'`	Válvula 4 (AIRE)
P9_21	motorPin	`motorPin = 'P9_21'`	Motor de succión
P8_11	Temp22	`Temp22 = 'P8_11'`	Sensor de temperatura/humedad
P9_12	sensorPin555	`sensorPin2600 = 'P9_12'`	Sensor TGS2600 a la salida del 555
P9_14	heatPin2600	`heatPin2600 = 'P9_14'`	Calentamiento del sensor TGS2600

Tabla 20: *Pines de conexión para experimentación de modulación en frecuencia por Martinelli.*

3. Conectar el transformador de 6V a la plataforma y el de 5V si no están conectados.

4. Situar los botes de odorante en su lugar y conectar el tubo de cada electroválvula al bote adecuado. Dependiendo del experimento a realizar igual se necesitan otros odorantes, por defecto se han utilizado metanol, etanol y butanol.

5. Situar el sensor de modulaciones en frecuencia en su posición dentro del circuito de adquisición. Para su rápida distinción se ha pintado una F en el sensor.

6. Ejecutar la sentencia de ejecución. Para el caso de la modulación en frecuencia por Martinelli es (véase el Anexo [H.2]):

```
tgs2600martinelli.py succion muestras switch
```

Donde *succión* es la velocidad de absorción del motor que tiene que tener un valor entre 50 y 100. *Switch* es el tiempo en segundos que se mantendrá cada electroválvula abierta y, por último, *tiempo* es la duración de la experimentación. Mientras que *muestras* se refiere a la cantidad de medidas a realizar. En este punto cabe especificar que tomamos como muestra todos los bloques de *K* pulsos que entre en *switch* segundos

7. La plataforma realizará la experimentación según lo explicado en la sección [5.3.3], el tiempo de duración del experimento será algo mayor del indicado por parámetro al ser la lectura asíncrona.

8. Tras finalizar la experimentación es conveniente apagar la plataforma si no se va a continuar utilizando. Para su desconexión realizar los siguientes pasos expuesto en el siguiente apartado.

6.4.1.1. Desconexión de la plataforma

1. Apagar la BBB mediante la sentencia: shutdown -h now

2. Esperar hasta que se apague la BBB, todos sus LEDs se apagarán.

3. Desconectar el transformador de 6V.

4. Desconectar el transformador de 5V y la conexión USB o Ethernet, en función de la conexión ssh que se haya realizado.

6.4.1.2. Ejemplo experimento Martinelli sin odorante

Se realizan los pasos indicados en la sección [6.4.16.4] hasta el punto 12. Como deseamos realizar una experimentación sin odorante nos cercioramos que no hay odorantes conectados al circuito y continuamos con paso 13. Se ejecuta la siguiente sentencia que inicia la experimentación.

```
tgs2600martinelli.py 100 10 100
```

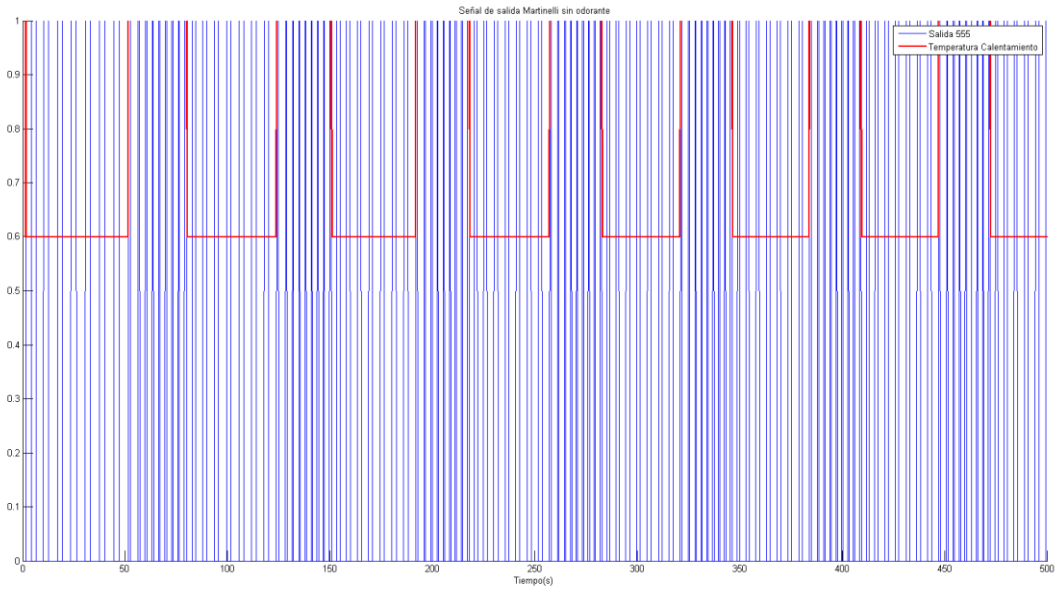


Figura 6. 9: *Salida experimentación Martinelli sin odorante. La señal azul es la respuesta del sensor TGS2600 a la salida del 555. La seña roja es la señal temperatura de calentamiento.*

Para la realización del experimento se ejecuta el script de martinelli con valor de velocidad de succión 100, es decir, succión máxima, tiempo de conmutación entre electroválvulas 100 segundos y se capturaran 10 muestras. En la gráfica de la Figura 6. 9 se puede observar la salida obtenida, se representan los primeros 600 segundos de la experimentación. Ahora realizaremos el mismo experimento, pero con odorante y se realizará una comparación de resultados con el fin de observar si ven diferencias.

6.4.1.3. Ejemplo experimento Martinelli con odorante

Se realizan los pasos indicados en la sección [6.4] hasta el punto 12. Como deseamos realizar una experimentación con odorante nos cercioramos que los odorantes estén conectados al circuito. En este caso vamos a realizar una conmutación de odorantes entre Metanol, Etanol y Aire. Aprovechamos la electroválvula 3 para el aire, dejándola libre, sin conectar ningún odorante.

Continuamos con paso 13 y se ejecuta la siguiente sentencia que inicia la experimentación.

```
tgs2600martinelli.py 100 10 100
```

Para la realización del experimento se ejecuta el script de regresión con valor de velocidad de succión 100, es decir, succión máxima, tiempo de conmutación entre electroválvulas 100 segundos y se capturaran 10 muestras. En la gráfica de la Figura 6. 10 se puede observar la salida obtenida, se representan los primeros 600 segundos de la experimentación.

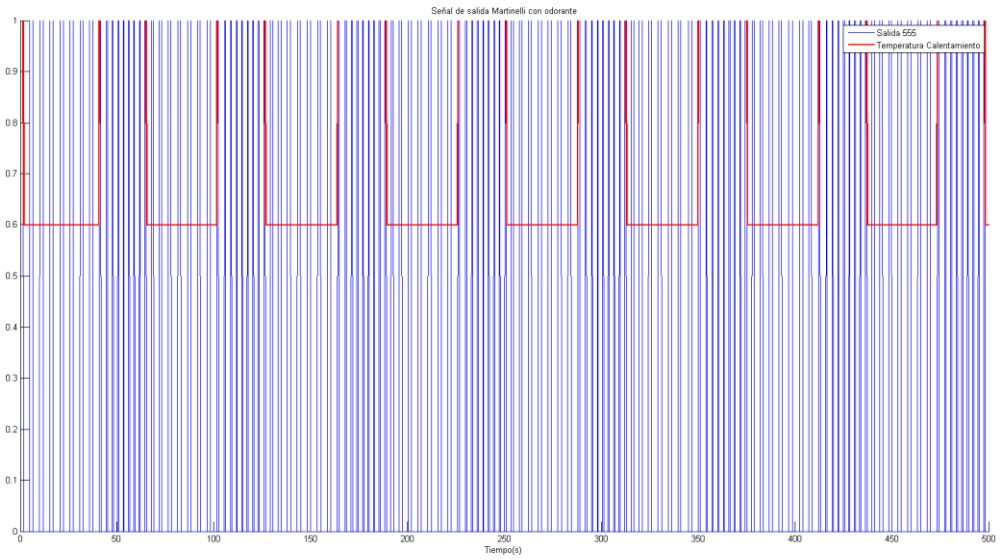


Figura 6. 10: *Salida experimentación Martinelli con odorante. La señal azul es la respuesta del sensor TGS2600 a la salida del 555. La seña roja es la señal temperatura de calentamiento.*

En este caso la experimentación se ha realizado con odorante y su vector de conmutación entre válvulas ha sido: Conmutación entre electroválvulas: [2 3 2 3 1 3 1 1 2 3]. Donde los

números 1, 2 y 3 son las electroválvulas (1-METANOL, 2-ETANOL, 3-BUTANOL, 4-AIRE).

Con el fin de hacer más fácil el análisis de los resultados se ha realizado un zoom de las gráficas de las dos experimentaciones entre 0 y 250 segundos de experimento.

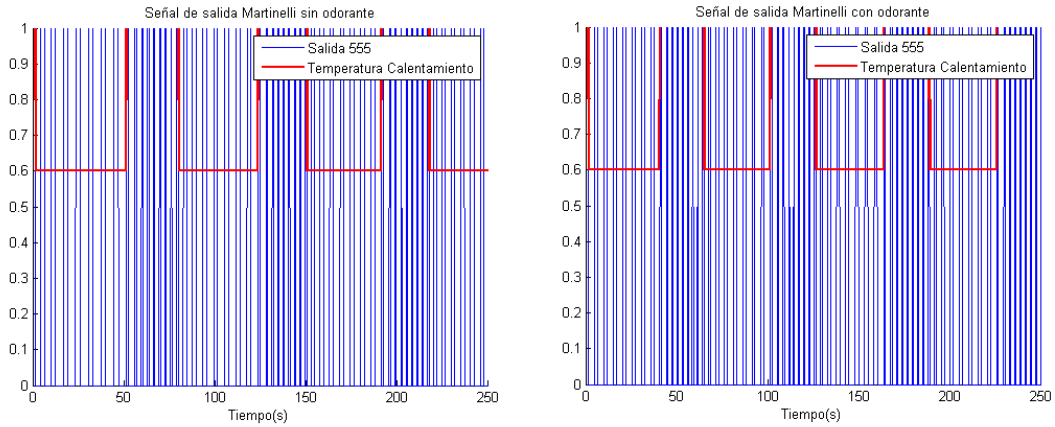


Figura 6. 11: *Experimentación Martinelli sin odorante, gráfica izquierda, frente a experimentación Martinelli con odorante, gráfica derecha.*

Como se puede observar en las gráficas a partir de la primera muestra (SAMPLESINICIO), que termina entorno a los 70 segundos. En presencia de odorante se produce una variación en la frecuencia de oscilación, observándose 7 transiciones de temperatura de calentamiento, mientras que, sin presencia de odorante, en el mismo tiempo, sólo hay 6 transiciones.

Para apreciar mejor la variación de frecuencia que se produce con cada odorante, se ha realizado una experimentación conmutado las válvulas por el siguiente orden: Metanol, Etanol y Butanol, con un valor de switch de 100 segundos. Los resultados se muestran en la Figura 6. 12.

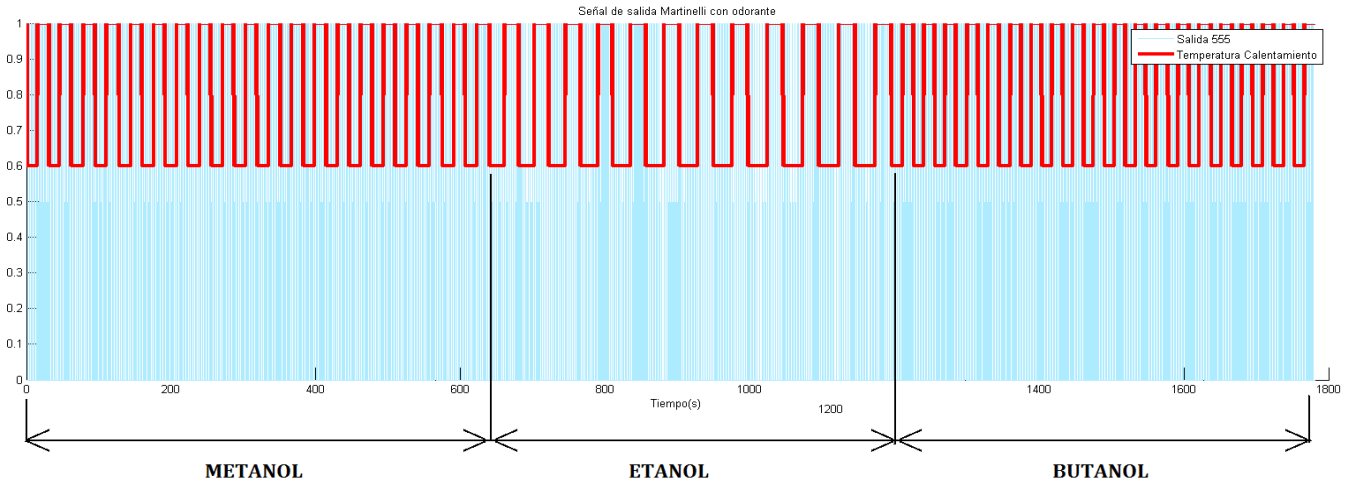


Figura 6. 12: *Experimentación Martinelli con Metanlo, Etanol y Butanol. Plumas de odorante de mínimo 600 segundos deduración.*

A la vista de la gráfica se aprecia que la frecuencia de oscilación es diferente para cada odorante, este hecho además se ha comprobado que se reproduce en otras experimentaciones. La mayor frecuencia se produce con el Butanol, pues en 600 segundos de conmutación de la electroválvula ha generado 20 transiciones de *k* pulsos de duración. El Etanol por lo contrario es el que menor frecuencia produce, generando 13 transiciones. Por el último el Metanol genera una frecuencia intermedia con 18 transiciones. En la Tabla 21 se muestran los resultados de forma numérica.

ODORANTE	MUESTRAS (nº transiciones de k pulsos)	Duración media de la Muestra (s)	Duración media de cada pulso (s)	Frecuencia de pulso (Hz)
METANOL	18	33,88	2,1175	0,472255018
ETANOL	13	47,656	4,124076923	0,242478503
BUTANOL	20	31,092	1,748925	0,571779808

Tabla 21: *Resultados de la experimentación*

7. Conclusiones y trabajo futuro

En este capítulo se expondrán las conclusiones que se han obtenido de la realización de este proyecto con relación a los objetivos inicialmente planteados de la sección [1.2], así como la consecución de los mismos a lo largo de este trabajo. Posteriormente se expondrá el trabajo futuro fruto de las reflexiones que han surgido en el transcurso y que podrán ser desarrolladas posteriormente partiendo con este proyecto como base.

7.1. Conclusiones

Este proyecto está enfocado a la consecución de una plataforma de experimentación versátil y fácilmente programable, que permita integrar y aplicar diferentes técnicas de modulación para la adquisición y detección de odorantes. Es por tanto el primer paso en la integración de diferentes técnicas de modulación sobre una misma plataforma de experimentación en el estudio de odorantes. Para su consecución se ha desarrollado un proyecto multidisciplinar, que abarca tareas que se engloban en los campos de la programación, instrumentación electrónica, teoría de circuitos, teoría de control, computación adaptativa y hasta algo de bricolaje.

Ha sido necesaria una ardua tarea de búsqueda de información, lectura y comprensión de documentación extensa relacionada con las narices artificiales con el fin de conseguir una plataforma funcional y que permita ser evolucionada. En trabajos anteriores del GNB, ya se generó una plataforma de experimentación [2], tal y como se referenció en la sección [2.3]. Dicha plataforma contaba con un microchip PIC como centro de control, lo que limitaba mucho la funcionalidad de la misma, cada vez que se quería realizar una experimentación era necesario correr el Firmware del PIC y programar sobre el mismo. La nueva plataforma rompe con esta limitación al incorporar como centro de control una BeagleBone Black (Industrial) que lleva integrado un sistema embebido Linux, y permite programar una gran variedad de lenguajes de alto nivel como Python, C, C++, java, etc.

La plataforma permite realizar experimentaciones con o sin modulación, y dentro de estas últimas es posible realizarlas sobre la amplitud o sobre la frecuencia. Para lograrlo ha sido necesario conocer perfectamente el funcionamiento de los sensores TGS2600, así como sus características y limitaciones. Esto junto con las limitaciones impuestas por la tarjeta BeagleBone Black en cuanto a tensiones y corrientes, máximas y mínimas de entrada o salida, han hecho de la creación de los circuitos de adaptación una ardua tarea.

La elaboración de los circuitos de acondicionamiento supuso la parte más compleja del proyecto, pues las limitaciones impuestas por BeagleBone (véase sección [3.2.3]) dieron lugar a la realización de tres circuitos distintos para controlar los sensores TGS2600 y otros

dos circuitos para el control de las electroválvulas y el control de succión. La lectura para modulaciones en amplitud se realiza sobre el puerto ADC de la tarjeta, limitado a 1,8V. la lectura para modulaciones en frecuencia se realiza a través de un puerto GPIO, limitado a 3,3V. Y la lectura sin modulación también por un puerto ADC. Esto, sumado a que el valor de voltaje entregado por el PWM es de 3,3V llevo a realizar los tres circuitos distintos explicados en las secciones [5.2.1][5.2.2][5.2.3]. el primero permite realizar lecturas de odorante sin realizar modulaciones sobre la temperatura de calentamiento. El segundo permite realizar capturas de odorante controlando la temperatura de calentamiento con modulaciones en frecuencia, para ello fue necesario limitar la salida del circuito 555 a un voltaje de 3.3V. y el tercero vale para realizar capturas de odorante controlando la temperatura de calentamiento con modulación en amplitud, en este caso fue necesario realizar un circuito de potencia para aumentar la señal PWM con la que se controla la temperatura, y un divisor de tensión para asegurarnos que la salida no supere los 1,8V.

Cabe destacar que durante la realización de estos circuitos se quemó la placa BeagleBone Black, y por ello se compró una nueva placa, la misma, pero en su variedad industrial. Se mejoraron los circuitos, introduciendo medidas de protección en las entradas y salidas de la placa, como divisores de tensión o transistores Darlington (véase la sección [4.3.4]).

7.1.1. Resumen de objetivos y resultados

A continuación, expondremos un resumen de los resultados obtenidos con relación a los objetos planteados en el capítulo uno.

1. Objetivo: *Estudio, análisis y comprensión de los trabajos previos realizados en el departamento del GNB.*

   El desarrollo de este objetivo quedo expuesto en el capítulo número 2.
   Se realizó una lectura y comprensión de los trabajos previos realizados en el laboratorio del GNB. Además de los diferentes protocolos de comunicación entre sensores y controlador.

2. Objetivo: *Lectura y estudio de la literatura sobre narices artificiales.*

   El desarrollo de este objetivo quedo expuesto en el capítulo 2.

3. Objetivo: *Búsqueda y estudio de documentación, trabajos y publicaciones realizadas sobre técnicas de modulación en narices artificiales.*

   El desarrollo de este objetivo quedo expuesto en el capítulo 2 y 3.
   Se realizó el estudio y comprensión de técnicas de adquisición y discriminación de odorantes tanto en amplitud como en frecuencia. Se elaboraron distintos circuitos de modulación, expuestos en el capítulo 5.

4. Objetivo: *Estudio general de sustancias a detectar*.

   El proyecto no se ha centrado en las sustancias a detectar, pero se ha comprobado el funcionamiento de la plataforma para los odorantes indicados en la sección

[6.1.1]. El estudio de las sustancias a detectar será uno de los posibles futuros trabajos a realizar.

5. Objetivo: *Elaboración de la plataforma de detección.*

   El desarrollo de este objetivo se expone en todos los capítulos de la memoria, se va a especificar por partes los resultados obtenidos para este objetivo.

   a. Subobjetivo: *Estudio y entendimiento de la plataforma de David Yañez [2].*

      A lo largo del capítulo 1 se exponen las motivaciones derivadas del estudio y entendimiento de la plataforma de David.

   b. Subobjetivo: *Estudio de los diferentes tipos de sensores disponibles en el mercado. Comparación entre ellos y elección del sensor más adecuado para la plataforma.*

      En el capítulo 3 se encuentra la información vinculada a este objetivo, siendo finalmente un TGS2600 el sensor utilizado.

   c. Subobjetivo: *Estudio y comparación entre las plataformas de hardware libre disponibles en el mercado (Arduino, Rapsberry y Beaglebone). Elección de la más adecuada para su integración en la plataforma.*

      El desarrollo de este objetivo se encuentra en el capítulo 3.
      Se realizó una comparación entre distintos sistemas de control disponibles en el mercado y se decidió utilizar una BBB.

   d. Subobjetivo: *Estudio y comparación de componentes para la realización de un circuito de adquisición de odorante.*

      En los capítulos 3 y 4 se encuentra la información relativa a la elección de componentes para el circuito de adquisición. Se realizó un circuito de flujo constante que permite la conmutación de cuatro electroválvulas, tres con odorante y una con aire.

   e. Subobjetivo: *Estudio y diseño de circuitos electrónicos para la adaptación de señales eléctricas.*

      En el capítulo 5 se ha documentado la realización de los circuitos de adaptación necesarios para la adquisición de señales con la BBB. Se han realizado tres circuitos uno para experimentaciones sin modulación, otro para experimentaciones por modulación en amplitud y el ultimo para experimentaciones con modulación en frecuencia.

   f. Subobjetivo: *Búsqueda y elección del resto de componentes a integrar en la plataforma. Condensadores, resistencias, electroválvulas, motor de succión, transistores, transformadores, etc.*

En el capítulo 5 se encuentran documentados los componentes secundarios necesarios para la realización de la plataforma. Han sido realizados dos circuitos de control, uno para el sistema de electroválvulas y otro para el control del motor.

g. Subobjetivo: *Estudio de la disposición de los componentes en la plataforma, con el fin de crear una plataforma de tamaño reducido, de fácil acceso a sus partes y con las menores distancias posibles entre componentes, para así reducir errores en las medidas.*
En el capítulo 5 se muestra como se ha realizado la estructura de la plataforma, y la integración de los componentes en la misma con el fin de reducir su tamaño al máximo y dejando un fácil acceso a sus partes.

6. Objetivo: *Estudio y familiarización con el entorno de programación de la plataforma elegida (Python). Desarrollo del software del sistema.*

   En el capítulo 4 se introducen las características principales del entorno de desarrollo y programación de la plataforma. A modo de ayuda sea creado una guía de uso de la plataforma para principiantes, se encuentra en el Anexo [E].

7. Objetivo: *Pruebas con la plataforma de detección y Estudio de los resultados obtenidos.*

   En el capítulo 6 se han realizado pruebas para los tres tipos de experimentaciones introducidas en la plataforma. Y se ha comprobado que el sistema funciona y está listo para utilizarse en futuros proyecto de discriminación y detección de odorantes.

7.2. Trabajo futuro

Este proyecto es el primer paso en la integración de diferentes técnicas de modulación sobre una misma plataforma de experimentación. Tras la elaboración de la misma y durante su desarrollo han surgido diferentes ideas para posibles mejoras que pueden desarrollarse en trabajos futuros. Aunque el principal trabajo futuro a realizar es la incorporación en la plataforma de algoritmos de clasificación y discriminación del odorante, realizando muchas y diversas experimentaciones para obtener patrones de discriminación. Unas mejoras van relacionadas con la parte software, mientras que otras se ligan al hardware, por ello se ha querido organizar en: mejoras estructurales, mejoras de circuitos y nuevas técnicas de modulación.

7.2.1. Mejoras estructurales

- Ampliación del array de electroválvulas para la incorporación de más odorantes. O apertura simultanea de odorante más aire para realizar mezclas.

- Realizar una nueva caja hermética donde se puedan incluir los tres sensores, y así poder realizar experimentaciones en paralelo.

7.2.2. Mejoras de circuitos

- Modularización de los diferentes circuitos del sistema por medio de PCB.

- Diseño en PCB de los siguientes circuitos:

    - Circuito de modulación en amplitud.
    - Circuito de modulación en frecuencia.
    - Circuito sin modulaciones.
    - Circuito control electroválvulas.
    - Circuito control motor.

7.2.3. Técnicas de modulación

- Modulaciones en Amplitud. Se ha incorporado un script para la modulación en amplitud por regresión de temperatura, pero se pueden realizar muchos otros tipos de experimentaciones si se actúa sobre la temperatura de calentamiento de distinta forma, como, por ejemplo:

    - Modulación lineal. Variando la temperatura de calentamiento de forma lineal.
    - Modulación sinusoidal. Haciendo que la temperatura siga una forma senoidal o cosenoidal.
    - Modulación dirigida por eventos externos en la señal de medida,
    - También sería factible integrar un controlador PID para forzar a que el sistema responda a un determinado tipo de función, lineal, senoidal, etc.

- Modulaciones en Frecuencia. Se ha integrado un script para la modulación en frecuencia por el método de Martinelli, este cambia la temperatura de calentamiento entre dos estados. Se pueden crear nuevas experimentaciones o modulaciones variando la forma en que se controla la temperatura. Además, sería recomendable hacer un estudio de esta modulación en función de la variación del parámetro k, ahora fijado en 16 pulsos. O modulaciones sujetas a eventos en la frecuencia de la señal medida.

- Modulaciones por Flujo. No era uno de los puntos del proyecto, pero se ha realizó el control del flujo de succión de forma que en futuros trabajos este parámetro pudiera ser una variable modulable. Así pues, adquiere importancia el realizar una modulación controlando el flujo de odorante que llega al sensor, pudiendo aumentar o disminuir el flujo en función de si se detecta mayor o menor cantidad de odorante.

Referencias

[1] David J. Yáñez Villarreal. Estrategias bioinspiradas para la adquisición de olores en narices artificiales. Master's thesis, Escuela Politécnica Superior, Universidad Autónoma de Madrid., 2009.

[2] David J. Yañez, Adolfo Toledano, Eduardo Serrano, Ana M. Martín de Rosales, Francisco B. Rodriguez and Pablo Varona. Characterization of a clinical olfactory test with an artificial nose. *Frontiers in neuroengineering,* 5, 2012.

[3] Fernando Herrero-Carrón, David J. Yáñez, Francisco de Borja Rodríguez and Pablo Varona. An active, inverse temperature modulation strategy for single sensor odorant classification. *Sensors and Actuators B: Chemical,* 2014.

[4] Carlos García-Saura, Francisco de Borja Rodríguez and Pablo Varona. Design principles for cooperative robots wit uncertainty-aware and resource-wise adaptive behavior. In *Biometic and Biohybrid sustems,* pages 108-117. Springer, 2014.

[5] Tomás Vázquez Rubio. Integración de una nariz electronica ultra-portátil en un robot modular para el control de su movimiento a través de los odorantes recibidos. Master's thesis, Escuela Politécnica Superior, Universidad Autónoma de Madrid., 2013.

[6] Carlos Garcia-Saura. Cooperative strategies for the detection and localization of odorants with robots and artificial noses. Master's thesis, Escuela Politécina Superios, Universidad Autónoma de Madrid., 2014.

[7] Alejandro Pequeño Zurro, *Uso de una nariz electronica ultra-portátil en robots para la detección de fuentes de odorante*s. Trabajo final de Carrera, Escuela Politécnica Superior, Universidad Autónoma de Madrid., 2015.

[8] Eugenio Martinelli, Davide Polese, Alexandro Catini, Arnaldo D'Amico, Corrado Di Natale. Self-adapted temperature modulation in metal-oxide semiconductor gas sensors. *Sensors and Actuators B: Chemical volume 161,* issue 1, 3 January 2012, pages 534-541.

[9] Eugenio Martinelli, Alexandro Catini, Corrado DI Natale. An active temperature modulation of gas sensor based on a selfadaptative strategy. *Solid-State Sensors, Actuators and Microsystems (TRADUCERS & EUROSENSORS XXVII)*, 213 Transducers & Eurosensors XXVII: The 17th International Conference on (IEEE).

[10] FIGARO GAS SENSOR TGS. Tgs2600 figaro engineering inc. *Japan (Nov. 1, 1978)*, 1978.

[11] T. C. Pearce, S. Schiffman, H. Troy Nagle & J. W. Gardner, *Handbook of Machine Olfaction: Electronic Nose Technology*, John Wiley & Sons 2003.

[12] R. Díaz Delgado, *Sensores de gases basados en óxidos de estaño: una aproximación electroquímica*, Tesis doctoral, Universidad de Barcelona, Barcelona, 2002.

[13] A. Heilig, N. Barsan, U. Weimar, M. Schweizer-Berberich, J. W. Gardner & W. Gopel, *Gas identification by modulating temperatures of SnO2-based thick film sensors*, Sensors and Actuators B-Chemical, 1997, 43, 45-51.

[14] T. Sahm, A. Gurlo, N. Barsan & U. Weimar, *Basics of oxygen and SnO2 interaction; work function change and conductivity measurements*, Sensors and Actuators B-Chemical, 2006, 118, 78-83.

[15] Gerald Coley of BeagleBoard.org. Beaglebone black system reference manual. Revision C.1. May 22, 2014

[16] Nanostructured Tin Dioxide Materials for Gas Sensor Applications, chapter 30. Functional Nanomaterials, 2006.

[17] NE555 Timer. Texas Instruments. http://www.ti.com/lit/ds/symlink/ne555.pdf

[18] Electrovalvula SY114-VLOZ-Q de la casa SMC. Datasheet: http://www.farnell.com/datasheets/309105.pdf

[19] R Chutia and M Bhuyan. *Study of temperature modulated tin oxide gas sensor and identification of chemicals*. In Computational Intelligence and Signal Processing (CISP), 2012 2nd National Conference on, pages 181-181. IEEE, 2012.

[20] R Gutierrez-Osuna, A Gutierrez-Galvez, and N Powar. *Transient response analysis for temperarture-modulated chemoresistors*. Sensors and Actuators B: Chemical, 93(1):57-66, 2003.

[21] Andrew P Lee and Brian J Reedy. *Temperature modulation in semiconductor gas sensing*. Sensors and Actuators B: Chemical, 60(1):35-42, 1999

[22] RS D200 Micropump Range, datasheet:

http://docs-europe.electrocomponents.com/webdocs/145e/0900766b8145ed36.pdf

[23] DHT22/AM2302 datasheet:

https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf

[24] Matt Richardson. *Getting started with BeagleBone*. Linux-Powered electronic projects with Python and JavaScript

[25] Gerald Coley. *BeagleBone Black System Reference Manual*.

[26] Derek Molloy. *Exploring BeagleBone tools and techniques for building with embebded linux*.

[27] Mark A. Yoder & Jason Kridner. *BeagleBone Cookbook, software and hardware problems and solutions*.

[28] Tutorial Python: http://docs.python.org.ar/tutorial/index.html

[29] Transistor PN2222A, datasheet:

https://www.fairchildsemi.com/datasheets/PN/PN2222A.pdf

[30] PN2222A transistor NPN de la casa Fairchild, datasheet:

https://www.fairchildsemi.com/datasheets/PN/PN2222A.pdf

[31] Kal Mustafa. Filtering techniques: Isolating analog and digital powersuplies in ti's pll-based ide devices. Technical report, Texas Instruments.

http://www.ti.com/lit/an/scaa048/scaa048.pdf

[32] ULN2003 de la fácbrica Texas Instruments, datasheet:

http://www.ti.com/lit/ds/symlink/uln2003a.pdf

[33] TIP120 transistor de la fábrica Faisrchild, datasheet:

https://www.fairchildsemi.com/datasheets/ti/tip120.pdf

[34] Figaro. Operating principle in mos type sensors. Technical report, Figaro.

http://www.figarosensor.com/technicalinfo/principle/mos-type.html

[35] FIGARO GAS SENSOR TGS. Tgs2620/tgs2611 figaro engineering inc. *Japan (Nov. 1, 1978)*, 1978.

[36] Diodo 1N4001, datasheet: http://www.onsemi.com/pub_link/Collateral/1N4001-D.PDF

Glosario

- BBB: BeagleBone Black.
- GNB: Grupo de Neurocomputacion Biológica.
- MOS: Metal Oxide Semiconductors.
- PWM: Pulse Wide Modulation.
- CP: Conductive Polymer.
- MOSFET: Sensor transistor de efecto de campo basado en MOS.
- QCM: Quartz MicroBalance.
- SAW: Surface Acustic Wave.
- EN: Electronic Noise.
- CVD: Chemical Vapor Deposition.
- CPU: Central Processing Unit.
- RAM: Random Access Memory.
- ROM: Read Only Memory.
- E/S: Entrada/Salida.
- IDE: Integrated Development Environment.
- GPIO: General Purpose Input/Output.
- EPROM: Erasable Programmable Read Only Memory.
- LDO: Low-DropOut.
- SPI: Serial Peripheral Interface.
- ADC: Analog to Digital Converter.
- CAN: Controller Area Network.
- TGS: Tagachi Gas Sensor.

- NC: Normally Close.
- SSH: Secure Shell.
- LEDs: Light-Emitting Diode.
- BIOS: Basic Input Output System.
- NTP: Network Time Protocol.
- GMT: Greenwich Mean Time.
- SO: Sistema operativo.
- OPKG: Open PacKaGe management.
- APT: Advanced Packaging Tool.
- DPKG: Debian PacKaGe.

Anexos

A. Sistemas embebidos.

Los sistemas embebidos se encuentran disponibles en cada momento de nuestra vida. El lavaplatos, el horno, el automóvil, el ascensor son controlados por computadoras, que por lo general no poseen pantalla, teclado o disco duro, y no responden a lo que comúnmente se denomina PC.

Aunque, normalmente, no son muy nombrados, los sistemas embebidos se encuentran en muchas partes. Es difícil encontrar un dispositivo, cuyo funcionamiento no esté basado en algún sistema embebido.

Estos sistemas suelen tener integrado en sus partes una computadora con unas características especiales conocida como microcontrolador, lo sería el cerebro del sistema. Se trata de un microprocesador con interfaces de entrada/salida en el mismo chip. Normalmente incluyen una interfaz externa para efectuar un monitoreo del estado y hacer un diagnóstico del sistema.

El diseño de un producto que incorpora sistemas embebidos generalmente está orientado a minimizar los costos y maximizar la confiabilidad. A menudo, estos sistemas operan en un ambiente dedicado con condiciones operacionales y escenarios muy específicos. Por lo que protocolos o funciones que son acertadas para un sistema de propósito general podrían no serlo para un sistema embebido.

Definición de sistema embebido.

Se entiende por sistema embebido a la combinación de software y hardware, con tal vez algunas piezas mecánicas o de otro tipo, diseñado para tener una función específica. Son dispositivos muy habituales pero que por lo general no dan la impresión de contar con un procesador y un programa ejecutándose que les permita funcionar.

Esto contrasta con respecto a una computadora personal, que, si bien también está compuesta por hardware y software, más algunas piezas mecánicas. Sin embargo, no está diseñada para un uso específico. Si no que es posible darle muchos usos diferentes.

Esta combinación hardware-software en muchos casos puede ser sustituida por un circuito integrado que realice la misma tarea. Pero una de las ventajas de los sistemas embebidos es su flexibilidad, ya que a la hora de realizar alguna modificación resulta mucho

más sencillo modificar líneas de código en el software del sistema embebido que reemplazar todo el circuito integrado.

Un uso muy común de los sistemas embebidos es en los sistemas de tiempo real, entendiéndose por sistemas en tiempo real a aquellos sistemas en los que el control del tiempo es vital para el correcto funcionamiento. Los sistemas en tiempo real necesitan realizar ciertas operaciones o cálculos en un límite de tiempo. Donde ese límite de tiempo resulta crucial. Un ejemplo claro de un sistema de tiempo real es el control de tráfico aéreo.

A.1. Estructura y componentes de un sistema embebido.

Las características principales de un sistema embebido son el bajo coste y consumo de potencia. Dado que muchos sistemas embebidos son producidos en miles o millones de unidades, el costo por unidad es un aspecto a tener en cuenta en su etapa de diseño. Generalmente estos sistemas cuentan con procesadores muy básicos, relativamente lentos y memorias pequeñas para minimizar costos.

En estos sistemas la velocidad no está ligada únicamente a la velocidad del reloj del procesador, sino que el total de la arquitectura se simplifica con el fin de reducir costos. Normalmente, se emplean periféricos controlados por interfaces seriales síncronas que son bastante más lentas que los periféricos empleados en un PC.

Como se ha indicado un sistema embebido suele enfrentarse a fuertes restricciones de recursos, por tanto, normalmente deberá hacer uso de sistemas operativos especiales, denominados de tiempo real. Estos deberán responder a estímulos externos con fuertes restricciones de tiempo. Se dice entonces que un sistema trabaja en tiempo real si la información después de la adquisición y tratamiento es todavía vigente. Es decir, en el caso de una información que llega de forma periódica, los tiempos de adquisición y tratamiento deben de ser inferiores al periodo de actualización de dicha información. Así, un sistema embebido puede ser o no de tiempo real en función de los requerimientos específicos de la aplicación que se desee implementar.

Los programas en estos sistemas se ejecutan minimizando los tiempos muertos y con fuertes limitaciones de hardware, ya que no suelen tener discos duros, ni teclados o monitores, una memoria flash reemplaza los discos y algunos botones y una pantalla LCD reemplazan los dispositivos de interfaz. No obstante, se ha evolucionado mucho en este tipo de sistemas y ya existen algunos que no portan este tipo de elementos de hardware, pero si permiten conectárselos.

El software que controla un dispositivo de hardware, por ejemplo, en una memoria ROM, Flash o un circuito integrado se conoce como Firmware. Típicamente la programación en este tipo de dispositivos se realiza en lenguaje ensamblador o en lenguaje C, actualmente se han desarrollado algunas máquinas virtuales y otros compiladores que permiten el diseño de programas más complejos. Además, se pueden encontrar otras herramientas como depuradores, simuladores, bases de datos, etc., para la programación de este tipo de sistemas.

A.1.1. Componentes.

Un sistema embebido en principio está formado por un microprocesador y un software que se ejecute sobre este. Sin embargo, este software necesitará sin duda un lugar donde poder guardarse para luego ser ejecutado por el procesador. Esto podría ser una memoria RAM o ROM. Todo sistema embebido necesita una cantidad determinada de memoria, la cual puede incluso encontrarse dentro del mismo chip del procesador. Además, normalmente, contará con una serie de salidas y entradas necesarias para comunicarse con el exterior.

En su memoria solo reside el programa destinado a gobernar una aplicación determinada. Sus líneas de entrada/salida soportan el conexionado de los sensores y actuadores del dispositivo a controlar y todos los recursos complementarios disponibles tienen como única función atender a sus requerimientos. Estas son por lo general las características que comparten todos los sistemas embebidos, todo lo demás será diferente y dependerá en particular de la finalidad del sistema.

Un PC embebido posee una arquitectura semejante a la de un PC. A continuación, se detallan brevemente sus elementos básicos:

- Microprocesador: encargado de realizar las operaciones principales del sistema. Ejecuta el código para realizar una determinada tarea y dirige el funcionamiento de los demás elementos.

- Memoria: En ella se almacena el código de los programas que el sistema puede ejecutar, así como los datos. Su característica principal es que debe tener un acceso de lectura y escritura lo más rápido posible para que el microprocesador no pierda tiempo en tareas que no son operacionales. Al ser volátil, el sistema requiere de un soporte donde se almacenen los datos incluso sin disponer de alimentación o energía.

- Caché: es una memoria más rápida que la principal en la que se almacenan los datos y el código accedido últimamente. Dado que habitualmente estos programas realizar microtareas repetitivas, la cache sirve para ahorrar tiempo. Hace que no sea necesario acceder a la memoria principal si el dato o instrucción ya se encuentra en cache.

- Disco duro: contiene la información no volátil. A diferencia de la memoria que es de estado sólido éste suele ser magnético. Pero su excesivo tamaño a veces lo hace inviable para PCs embebidos, con lo que se requieren soluciones como discos de estado sólido. Existen en el mercado varias soluciones de esta clase (DiskOnChip, CompactFlash, IDE Flash Drive, etc.) con capacidades suficientes para la mayoría de sistemas embebidos (desde 2 hasta más de 1 GB). El controlador del disco duro de PCs estándar cumple con el estándar IDE y es un chip más de la placa madre.

- BIOS-ROM: BIOS (Basic Input & Output System, Sistema básico de entrada y salida) es código que es necesario para inicializar el ordenador y para poner en comunicación los distintos elementos de la placa madre. La ROM (Read Only

Memory, memoria de sólo lectura no volátil) es un chip donde se encuentra el código BIOS.

- CMOS-RAM: Es un chip de memoria de lectura y escritura alimentado con una pila donde se almacena el tipo y ubicación de los dispositivos conectados a la placa madre (disco duro, puertos de entrada y salida, etc.). Además, contiene un reloj en permanente funcionamiento que ofrece al sistema la fecha y la hora.

- Chip set: Chip que se encarga de controlar las interrupciones dirigidas al microprocesador, el acceso directo a memoria (DMA) y al bus ISA, además de ofrecer temporizadores, etc. Es frecuente encontrar la CMOS-RAM y el reloj de tiempo real en el interior del Chip set.

- Entradas y salidas: periféricos que permiten la comunicación entre un sistema de procesamiento de información y el exterior.

B. Placa BeagleBone Black

B.1. BBB. Pines de conexión y puertos de expansión

PIN	PROC	NAME	MODE0	MODE1	MODE2	MODE3
1,2						GND
3	R9	GPIO1_6	gpmc_ad6	mmc1_dat6		
4	T9	GPIO1_7	gpmc_ad7	mmc1_dat7		
5	R8	GPIO1_2	gpmc_ad2	mmc1_dat2		
6	T8	GPIO1_3	gpmc_ad3	mmc1_dat3		
7	R7	TIMER4	gpmc_advn_ale		timer4	
8	T7	TIMER7	gpmc_oen_ren		timer7	
9	T6	TIMER5	gpmc_be0n_cle		timer5	
10	U6	TIMER6	gpmc_wen		timer6	
11	R12	GPIO1_13	gpmc_ad13	lcd_data18	mmc1_dat5	mmc2_dat1
12	T12	GPIO1_12	gpmc_ad12	Lcd_data19	mmc1_dat4	Mmc2_dat0
13	T10	EHRPWM2B	gpmc_ad9	lcd_data22	mmc1_dat1	mmc2_dat5
14	T11	GPIO0_26	gpmc_ad10	lcd_data21	mmc1_dat2	mmc2_dat6
15	U13	GPIO1_15	gpmc_ad15	lcd_data16	mmc1_dat7	mmc2_dat3
16	V13	GPIO1_14	gpmc_ad14	lcd_data17	mmc1_dat6	mmc2_dat2
17	U12	GPIO0_27	gpmc_ad11	lcd_data20	mmc1_dat3	mmc2_dat7
18	V12	GPIO2_1	gpmc_clk_mux0	lcd_memory_clk	gpmc_wait1	mmc2_clk
19	U10	EHRPWM2A	gpmc_ad8	lcd_data23	mmc1_dat0	mmc2_dat4
20	V9	GPIO1_31	gpmc_csn2	gpmc_be1n	mmc1_cmd	
21	U9	GPIO1_30	gpmc_csn1	gpmc_clk	mmc1_clk	
22	V8	GPIO1_5	gpmc_ad5	mmc1_dat5		
23	U8	GPIO1_4	gpmc_ad4	mmc1_dat4		
24	V7	GPIO1_1	gpmc_ad1	mmc1_dat1		
25	U7	GPIO1_0	gpmc_ad0	mmc1_dat0		
26	V6	GPIO1_29	gpmc_csn0			
27	U5	GPIO2_22	lcd_vsync	gpmc_a8		
28	V5	GPIO2_24	lcd_pclk	gpmc_a10		
29	R5	GPIO2_23	lcd_hsync	gpmc_a9		
30	R6	GPIO2_25	lcd_ac_bias_en	gpmc_a11		
31	V4	UART5_CTSN	lcd_data14	gpmc_a18	eQEP1_index	mcasp0_axr1
32	T5	UART5_RTSN	lcd_data15	gpmc_a19	eQEP1_strobe	mcasp0_ahclkx
33	V3	UART4_RTSN	lcd_data13	gpmc_a17	eQEP1B_in	mcasp0_fsr
34	U4	UART3_RTSN	lcd_data11	gpmc_a15	ehrpwm1B	mcasp0_ahclkr
35	V2	UART4_CTSN	lcd_data12	gpmc_a16	eQEP1A_in	mcasp0_aclkr
36	U3	UART3_CTSN	lcd_data10	gpmc_a14	ehrpwm1A	mcasp0_axr0
37	U1	UART5_TXD	lcd_data8	gpmc_a12	ehrpwm1_tripzone_in	mcasp0_aclkx
38	U2	UART5_RXD	lcd_data9	gpmc_a13	ehrpwm0_synco	mcasp0_fsx
39	T3	GPIO2_12	lcd_data6	gpmc_a6		eQEP2_index
40	T4	GPIO2_13	lcd_data7	gpmc_a7		eQEP2_strobe
41	T1	GPIO2_10	lcd_data4	gpmc_a4		eQEP2A_in
42	T2	GPIO2_11	lcd_data5	gpmc_a5		eQEP2B_in
43	R3	GPIO2_8	lcd_data2	gpmc_a2		ehrpwm2_tripzone_in
44	R4	GPIO2_9	lcd_data3	gpmc_a3		ehrpwm0_synco
45	R1	GPIO2_6	lcd_data0	gpmc_a0		ehrpwm2A
46	R2	GPIO2_7	lcd_data1	gpmc_a1		ehrpwm2B

Figura B. 1: *Pines de conexión. Modos de funcionamiento Puerto 8 (P8). Parte 1.*

PIN 1,2	PROC	NAME	MODE4	MODE5	MODE6	MODE7
3	R9	GPIO1_6				gpio1[6]
4	T9	GPIO1_7				gpio1[7]
5	R8	GPIO1_2				gpio1[2]
6	T8	GPIO1_3				gpio1[3]
7	R7	TIMER4				gpio2[2]
8	T7	TIMER7				gpio2[3]
9	T6	TIMER5				gpio2[5]
10	U6	TIMER6				gpio2[4]
11	R12	GPIO1_13	eQEP2B_in	.	pr1_pru0_pru_r30_15	gpio1[13]
12	T12	GPIO1_12	Eqep2a_in		pr1_pru0_pru_r30_14	gpio1[12]
13	T10	EHRPWM2B	ehrpwm2B			gpio0[23]
14	T11	GPIO0_26	ehrpwm2_tripzone_in			gpio0[26]
15	U13	GPIO1_15	eQEP2_strobe		pr1_pru0_pru_r31_15	gpio1[15]
16	V13	GPIO1_14	eQEP2_index		pr1_pru0_pru_r31_14	gpio1[14]
17	U12	GPIO0_27	ehrpwm0_synco			gpio0[27]
18	V12	GPIO2_1			mcasp0_fsr	gpio2[1]
19	U10	EHRPWM2A	ehrpwm2A			gpio0[22]
20	V9	GPIO1_31		pr1_pru1_pru_r30_13	pr1_pru1_pru_r31_13	gpio1[31]
21	U9	GPIO1_30		pr1_pru1_pru_r30_12	pr1_pru1_pru_r31_12	gpio1[30]
22	V8	GPIO1_5				gpio1[5]
23	U8	GPIO1_4				gpio1[4]
24	V7	GPIO1_1				gpio1[1]
25	U7	GPIO1_0				gpio1[0]
26	V6	GPIO1_29				gpio1[29]
27	U5	GPIO2_22		pr1_pru1_pru_r30_8	pr1_pru1_pru_r31_8	gpio2[22]
28	V5	GPIO2_24		pr1_pru1_pru_r30_10	pr1_pru1_pru_r31_10	gpio2[24]
29	R5	GPIO2_23		pr1_pru1_pru_r30_9	pr1_pru1_pru_r31_9	gpio2[23]
30	R6	GPIO2_25				gpio2[25]
31	V4	UART5_CTSN	uart5_rxd		uart5_ctsn	gpio0[10]
32	T5	UART5_RTSN	mcasp0_axr3		uart5_rtsn	gpio0[11]
33	V3	UART4_RTSN	mcasp0_axr3		uart4_rtsn	gpio0[9]
34	U4	UART3_RTSN	mcasp0_axr2		uart3_rtsn	gpio2[17]
35	V2	UART4_CTSN	mcasp0_axr2		uart4_ctsn	gpio0[8]
36	U3	UART3_CTSN			uart3_ctsn	gpio2[16]
37	U1	UART5_TXD	uart5_txd		uart2_ctsn	gpio2[14]
38	U2	UART5_RXD	uart5_rxd		uart2_rtsn	gpio2[15]
39	T3	GPIO2_12		pr1_pru1_pru_r30_6	pr1_pru1_pru_r31_6	gpio2[12]
40	T4	GPIO2_13	pr1_edio_data_out7	pr1_pru1_pru_r30_7	pr1_pru1_pru_r31_7	gpio2[13]
41	T1	GPIO2_10		pr1_pru1_pru_r30_4	pr1_pru1_pru_r31_4	gpio2[10]
42	T2	GPIO2_11		pr1_pru1_pru_r30_5	pr1_pru1_pru_r31_5	gpio2[11]
43	R3	GPIO2_8		pr1_pru1_pru_r30_2	pr1_pru1_pru_r31_2	gpio2[8]
44	R4	GPIO2_9		pr1_pru1_pru_r30_3	pr1_pru1_pru_r31_3	gpio2[9]
45	R1	GPIO2_6		pr1_pru1_pru_r30_0	pr1_pru1_pru_r31_0	gpio2[6]
46	R2	GPIO2_7		pr1_pru1_pru_r30_1	pr1_pru1_pru_r31_1	gpio2[7]

Figura B. 2: *Pines de conexión. Modos de funcionamiento Puerto 8 (P8). Parte 2.*

PIN	PROC	NAME	MODE0	MODE1	MODE2	MODE3
1,2						GND
3,4						DC_3.3V
5,6						VDD_5V
7,8						SYS_5V
9						PWR_BUT
10	A10					SYS_RESETn
11	T17	UART4_RXD	gpmc_wait0	mii2_crs	gpmc_csn4	rmii2_crs_dv
12	U18	GPIO1_28	gpmc_be1n	mii2_col	gpmc_csn6	mmc2_dat3
13	U17	UART4_TXD	gpmc_wpn	mii2_rxerr	gpmc_csn5	rmii2_rxerr
14	U14	EHRPWM1A	gpmc_a2	mii2_txd3	rgmii2_td3	mmc2_dat1
15	R13	GPIO1_16	gpmc_a0	gmii2_txen	rmii2_tctl	mii2_txen
16	T14	EHRPWM1B	gpmc_a3	mii2_txd2	rgmii2_td2	mmc2_dat2
17	A16	I2C1_SCL	spi0_cs0	mmc2_sdwp	I2C1_SCL	ehrpwm0_synci
18	B16	I2C1_SDA	spi0_d1	mmc1_sdwp	I2C1_SDA	ehrpwm0_tripzone
19	D17	I2C2_SCL	uart1_rtsn	timer5	dcan0_rx	I2C2_SCL
20	D18	I2C2_SDA	uart1_ctsn	timer6	dcan0_tx	I2C2_SDA
21	B17	UART2_TXD	spi0_d0	uart2_txd	I2C2_SCL	ehrpwm0B
22	A17	UART2_RXD	spi0_sclk	uart2_rxd	I2C2_SDA	ehrpwm0A
23	V14	GPIO1_17	gpmc_a1	gmii2_rxdv	rgmii2_rxdv	mmc2_dat0
24	D15	UART1_TXD	uart1_txd	mmc2_sdwp	dcan1_rx	I2C1_SCL
25	A14	GPIO3_21*	mcasp0_ahclkx	eQEP0_strobe	mcasp0_axr3	mcasp1_axr1
26	D16	UART1_RXD	uart1_rxd	mmc1_sdwp	dcan1_tx	I2C1_SDA
27	C13	GPIO3_19	mcasp0_fsr	eQEP0B_in	mcasp0_axr3	mcasp1_fsx
28	C12	SPI1_CS0	mcasp0_ahclkr	ehrpwm0_synci	mcasp0_axr2	spi1_cs0
29	B13	SPI1_D0	mcasp0_fsx	ehrpwm0B		spi1_d0
30	D12	SPI1_D1	mcasp0_axr0	ehrpwm0_tripzone		spi1_d1
31	A13	SPI1_SCLK	mcasp0_aclkx	ehrpwm0A		spi1_sclk
32						VADC
33	C8					AIN4
34						AGND
35	A8					AIN6
36	B8					AIN5
37	B7					AIN2
38	A7					AIN3
39	B6					AIN0
40	C7					AIN1
41#	D14	CLKOUT2	xdma_event_intr1		tclkin	clkout2
	D13	GPIO3_20	mcasp0_axr1	eQEP0_index		Mcasp1_axr0
42@	C18	GPIO0_7	eCAP0_in_PWM0_out	uart3_txd	spi1_cs1	pr1_ecap0_ecap_capin_apwm_o
	B12	GPIO3_18	Mcasp0_aclkr	eQEP0A_in	Mcaspo_axr2	Mcasp1_aclkx
43-46						GND

Figura B. 3: *Pines de conexión. Modos de funcionamiento Puerto 9 (P9). Parte 1.*

PIN	PROC	NAME	MODE4	MODE5	MODE6	MODE7
1,2						
3,4						
5,6						
7,8						
9						
10	A10					
11	T17	UART4_RXD	mmc1_sdcd		uart4_rxd_mux2	gpio0[30]
12	U18	GPIO1_28	gpmc_dir		mcasp0_aclkr_mux3	gpio1[28]
13	U17	UART4_TXD	mmc2_sdcd		uart4_txd_mux2	gpio0[31]
14	U14	EHRPWM1A	gpmc_a18		ehrpwm1A_mux1	gpio1[18]
15	R13	GPIO1_16	gpmc_a16		ehrpwm1_tripzone_input	gpio1[16]
16	T14	EHRPWM1B	gpmc_a19		ehrpwm1B_mux1	gpio1[19]
17	A16	I2C1_SCL	pr1_uart0_txd			gpio0[5]
18	B16	I2C1_SDA	pr1_uart0_rxd			gpio0[4]
19	D17	I2C2_SCL	spi1_cs1	pr1_uart0_rts_n		gpio0[13]
20	D18	I2C2_SDA	spi1_cs0	pr1_uart0_cts_n		gpio0[12]
21	B17	UART2_TXD	pr1_uart0_rts_n		EMU3_mux1	gpio0[3]
22	A17	UART2_RXD	pr1_uart0_cts_n		EMU2_mux1	gpio0[2]
23	V14	GPIO1_17	gpmc_a17		ehrpwm0_synco	gpio1[17]
24	D15	UART1_TXD		pr1_uart0_txd	pr1_pru0_pru_r31_16	gpio0[15]
25	A14	GPIO3_21*	EMU4_mux2	pr1_pru0_pru_r30_7	pr1_pru0_pru_r31_7	gpio3[21]
26	D16	UART1_RXD		pr1_uart0_rxd	pr1_pru1_pru_r31_16	gpio0[14]
27	C13	GPIO3_19	EMU2_mux2	pr1_pru0_pru_r30_5	pr1_pru0_pru_r31_5	gpio3[19]
28	C12	SPI1_CS0	eCAP2_in_PWM2_out	pr1_pru0_pru_r30_3	pr1_pru0_pru_r31_3	gpio3[17]
29	B13	SPI1_D0	mmc1_sdcd_mux1	pr1_pru0_pru_r30_1	pr1_pru0_pru_r31_1	gpio3[15]
30	D12	SPI1_D1	mmc2_sdcd_mux1	pr1_pru0_pru_r30_2	pr1_pru0_pru_r31_2	gpio3[16]
31	A13	SPI1_SCLK	mmc0_sdcd_mux1	pr1_pru0_pru_r30_0	pr1_pru0_pru_r31_0	gpio3[14]
32						
33	C8					
34						
35	A8					
36	B8					
37	B7					
38	A7					
39	B6					
40	C7					
41#	D14	CLKOUT2	timer7_mux1	pr1_pru0_pru_r31_16	EMU3_mux0	gpio0[20]
	D13	GPIO3_20	emu3	pr1_pru0_pru_r30_6	pr1_pru0_pru_r31_6	gpio3[20]
42@	C18	GPIO0_7	spi1_sclk	mmc0_sdwp	xdma_event_intr2	gpio0[7]
	B12	GPIO3_18		pr1_pru0_pru_r30_4	pr1_pru0_pru_r31_4	gpio3[18]
43-46						

Figura B. 4: *Pines de conexión. Modos de funcionamiento Puerto 9 (P9). Parte 2.*

C. Fundamentos de Multivibradores

En electrónica un multivibrador es un circuito oscilador capaz de generar una onda cuadrada. Según su funcionamiento los multivibradores se pueden clasificar en dos clases:

- **Astable**. También conocido como de funcionamiento continuo o de oscilación libre. Genera ondas a partir de la propia fuente de alimentación.

- De funcionamiento impulsado. A partir de una señal de disparo o impulso sale de su estado de reposo. Se distinguen dos subclases:

    - Biestable. Si posee dos de dichos estados.
    - Monoestable. Si posee uno.

En su forma más sencilla son dos sencillos transistores realimentados entre sí. Usando redes de resistencias y condensadores en esa realimentación se puede definir los periodos de inestabilidad.

Un circuito integrado multivibrador muy popular es el 555, que usa un sofisticado diseño para lograr una gran precisión y flexibilidad con muy pocos componentes externos.

C.1. Multivibradores monoestables

El circuito electrónico que más se utiliza tanto en la industria como en los circuitos comerciales, es el circuito temporizador o de retardo. Un temporizador, básicamente, consiste en un elemento que se activa y desactiva después de un tiempo más o menos preestablecido.

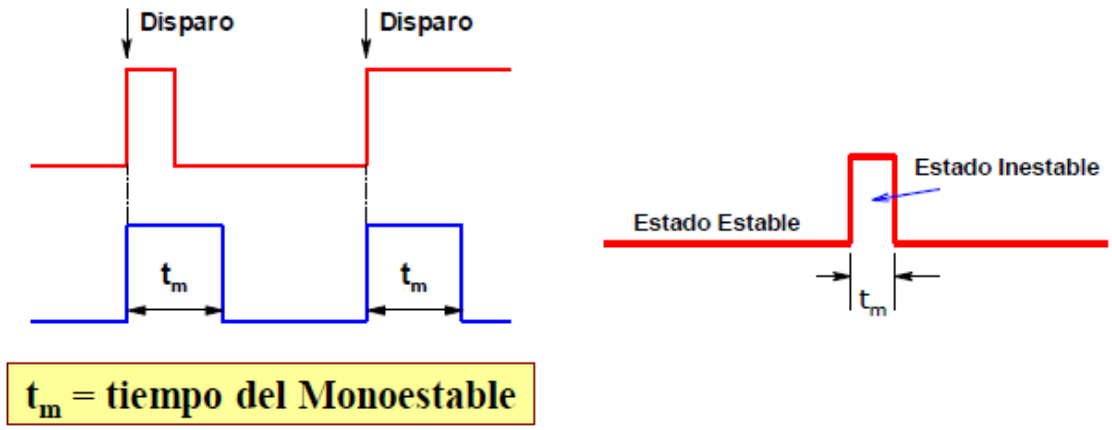


Figura C. 1: *Onda cuadrada generada por un multivibrador monoestable.*

De esta manera, se puede determinar el tiempo que ha de transcurrir para que el circuito susceptible a la temporización se detenga, empiece a funcionar o simplemente cierre un contacto o lo abra.

Un multivibrador monoestable posee dos estados, uno estable y otro inestable. A la vista de la Figura B.1, se pueden distinguir estos estados, cada vez que se da un pulso de disparo la salida del multivibrador para a estado alto por un tiempo t_m . El tiempo t_m es independiente del ancho del pulso. Si durante el tiempo t_m hay otro disparo este se suma si el monoestable es redisparable.

Un circuito integrado multivibrador muy popular es el 555, que usa un sofisticado diseño para lograr una gran precisión y flexibilidad con muy pocos componentes externos. En la siguiente figura se puede observar la circuitería interna de dicho integrado.

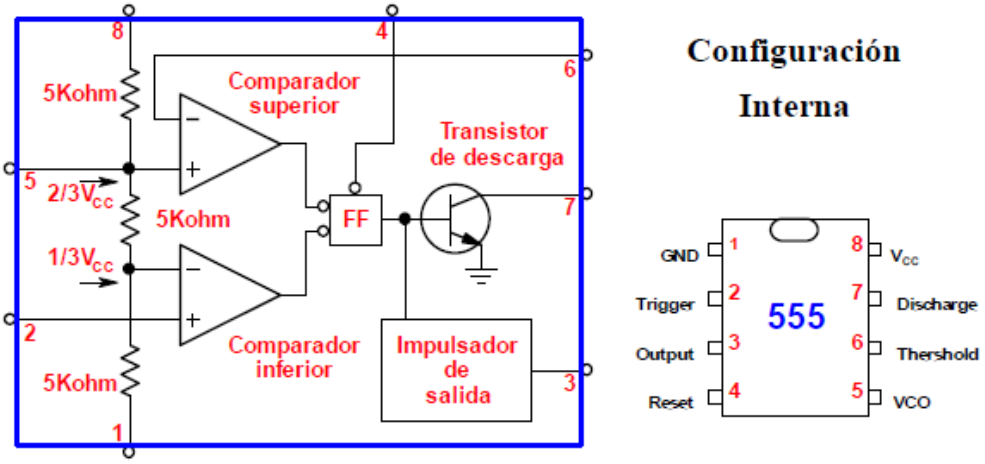


Figura C. 2: *Configuración interna 555.*

Y una configuración típica del 555 como multivibrador monoestable es la siguiente:

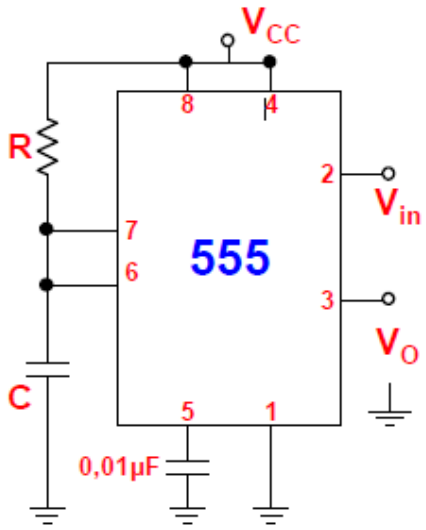


Figura C. 3: *Multivibrador monoestable con circuito integrado 555.*

Donde el tiempo del monoestable viene dado por: $t_m = 1.1 \times R_A \times C$

C.2. Multivibradores astables

Se trata de multivibradores que no tienen ningún estado estable, es decir, posee dos estados "casi-estables" entre los que conmuta, permaneciendo en cada uno de ellos un tiempo determinado. La frecuencia de la conmutación depende, en general, de la carga y la descarga de condensadores. Entre sus múltiples aplicaciones se encuentran la generación de ondas periódicas (generador de reloj) y de trenes de pulsos. Puede estar formado con amplificadores operacionales, puertas lógicas, circuitos integrados…

Figura C. 4: *Onda cuadrada generada por un multivibrador astable.*

A la vista de la figura, es deducible, que en este caso habrá dos tiempos distinto. El tiempo de encendido (t_{ON}) y el tiempo de apagado (t_{OFF}). Se puede definir entonces el ciclo de trabajo o duty cicle, como:

$$D = \frac{t_{ON}}{t_{ON} + t_{OFF}}$$

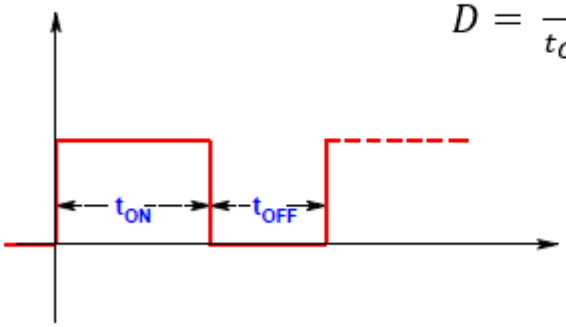


Figura C. 5: *Onda cuadrada generada por multivibrador astable. Ciclo de trabajo.*

En el caso de los multivibradores astables, el circuito integrado 555, también es una de las opciones más utilizadas. Siendo su configuración la presentada en la figura B.6.

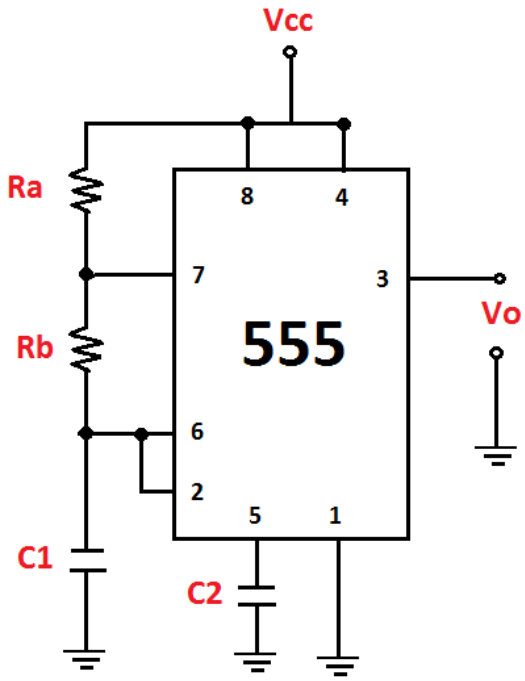


Figura C. 6: *Multivibrador astable con circuito integrado 555.*

En este caso, el cálculo de los tiempos de subida y bajada se complican un poco en comparación con la configuración monoestable. Estos tiempos van a depender del valor de los componentes externos del circuito. Analizando el circuito de la figura se obtienen las siguientes ecuaciones para los tiempos de encendido y apagado.

$$t_{ON} = 0.693 \times (R_A + R_B) \times C$$

$$t_{OFF} = 0.693 \times R_B \times C$$

Donde t_{ON}, se refiere al tiempo de encendido y corresponde al tiempo de carga del condensador C. Y donde t_{OFF}, se refiere al tiempo de apagado y corresponde al tiempo de descarga del condensador C.

El tiempo total vendrá dado por: $T = t_{ON} + t_{OFF} = 0.693 \times (R_A + 2R_B) \times C$

Luego si se quiere obtener $t_{ON} = t_{OFF}$, implica que $R_B \gg R_A$

$$T = 1.38 \times R_B \times C$$

D. Fundamentos de Electroválvulas

Una electroválvula, o válvula solenoide, es un dispositivo electromecánico operado eléctricamente, y es utilizado para controlar el flujo de líquidos o gases en posición completamente abierta o completamente cerrada. No se debe confundir la electroválvula con válvulas motorizadas, que son aquellas en las que un motor acciona el cuerpo de la válvula.

Este tipo de válvula, se cierra por gravedad, por presión o por la acción de un resorte; y es abierta por el movimiento de un émbolo operado por la acción magnética de una bobina energizada eléctricamente, o viceversa. El término solenoide no se refiere a la válvula misma, sino a la bobina montada sobre la válvula, con frecuencia llamada operador. La palabra "solenoide" se deriva de las palabras griegas "solen", que significa canal, y "oide" que significa forma. La bobina proporciona un canal, en el cual se crea una fuerte fuerza magnética al energizar la bobina.

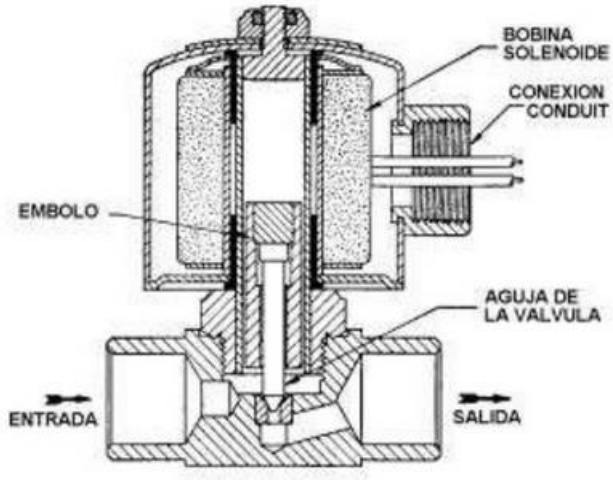


Figura D. 1: *Estructura interna válvula de solenoide de acción directa.*

La válvula de solenoide pude usarse para controlar el flujo de muchos fluidos diferentes, dándole la debida consideración a las presiones y temperaturas involucradas, la viscosidad del fluido y la adaptabilidad de los materiales usados en la construcción de la válvula.

Existe una amplia variedad de tipos de válvula solenoide, los cuales se pueden dividir de acuerdo a su aplicación, su construcción y su forma. Atendiendo a su aplicación, se pueden dividir de manera general, en dos tipos:

- Acción directa.
- Operadas por piloto.

Solenoide de acción directa: Se utiliza en válvulas con baja capacidad y puertos de tamaño pequeño. El émbolo está conectado mecánicamente a la aguja de la válvula. Al energizar la bobina, el émbolo se eleva hacia el centro de la misma, levantando la aguja. La **¡Error! No se encuentra el origen de la referencia.** muestra una típica configuración de e ste tipo de válvulas.

Solenoide de acción pilotada: son ideales para presiones y/o flujos mayores. Sustituyen la acción directa apoyándose en la presión de línea para abrir orificios más grandes manteniendo el tamaño del solenoide pequeño. En estas válvulas, el émbolo está unido a un vástago de aguja que cubre un orificio piloto en lugar del puerto principal. En algunos diseños de válvulas de solenoide operadas por piloto, se usa un diafragma en lugar de pistón, para cerrar el puerto principal. En la figura de abajo se muestra la estructura de una de este tipo.

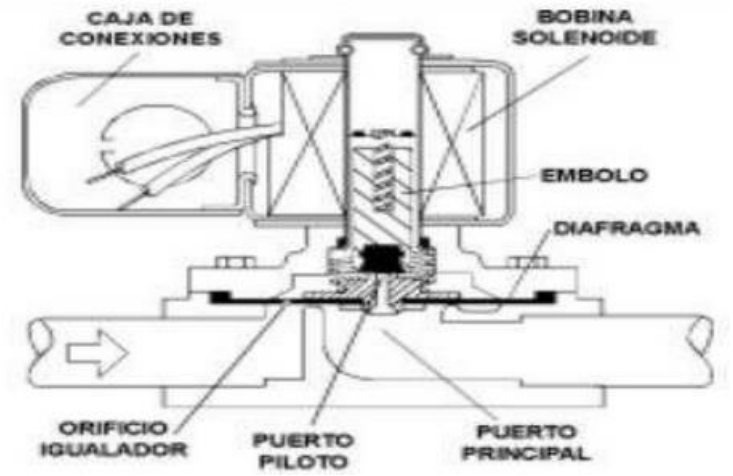


Figura D. 2: *Estructura interna válvula de solenoide de acción pilotada.*

Por su construcción, las válvulas solenoides pueden ser:

- Normalmente cerradas
  .
- Normalmente abiertas.

Solenoide normalmente cerrado (N.C.): abren cuando son energizadas y cierran cuando no son energizadas.

Solenoide normalmente abierto (N.O o N.A.): cierran cuando son energizadas y abren cuando no son energizadas.

Por su forma, hay tres tipos de válvulas solenoides de uso común:

- De dos vías.
- De tres vías.
- De cuatro vías o reversibles.

Solenoide de 2 vías: es el tipo de válvulas más común, tiene una conexión de entrada y una de salida y controla el flujo del fluido en una sola línea. Se utilizan para permitir e interrumpir el flujo de fluidos. Los dos funcionamientos se denominan, normalmente cerrado y normalmente abierto. Se debe instalar para trabajar en la dirección indicada por el fabricante ya que el no hacerlo resultará en operación ineficiente o imposible.

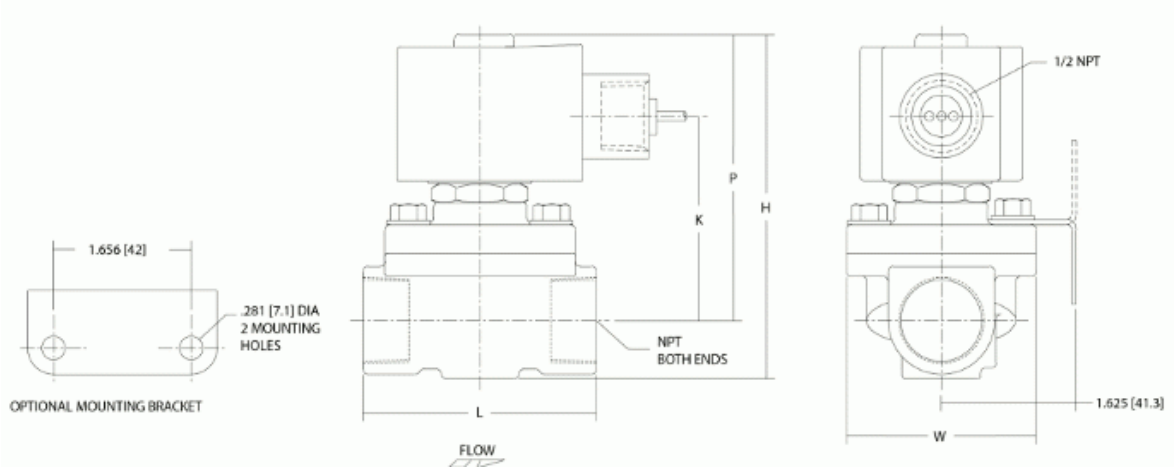


Figura D. 3: *Diagrama de flujo electroválvula de 2 vías y 2 posiciones.*

Solenoide de 3 vías: estas válvulas tienen tres conexiones, una de entrada que es común a dos diferentes conexiones de salida. Cuando se abre un orificio, el otro se cierra, y viceversa. Son, básicamente, una combinación de la válvula de dos vías, en un solo cuerpo y con una sola bobina.

Estas válvulas suelen usarse para aplicar o quitar presión a un actuador de válvula o a un cilindro de efecto simple. Estas válvulas pueden ser normalmente cerradas, normalmente abiertas o universales. La mayoría de ellas son operadas por piloto.

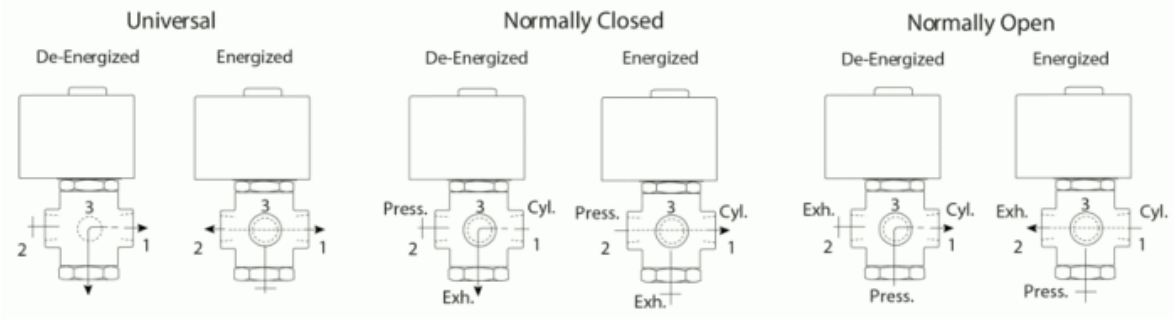


Figura D. 4: *Diagrama de flujo electroválvula de 3 vías y 2 posiciones.*

Solenoide de 4 vías: estas válvulas tienen cuatro o cinco conexiones, normalmente llamadas puertos. Cuentan con una entrada de presión, dos puertos cilíndricos que proveen presión a un cilindro o actuador de efecto doble, y una o dos salidas para liberar presión de los cilindros.

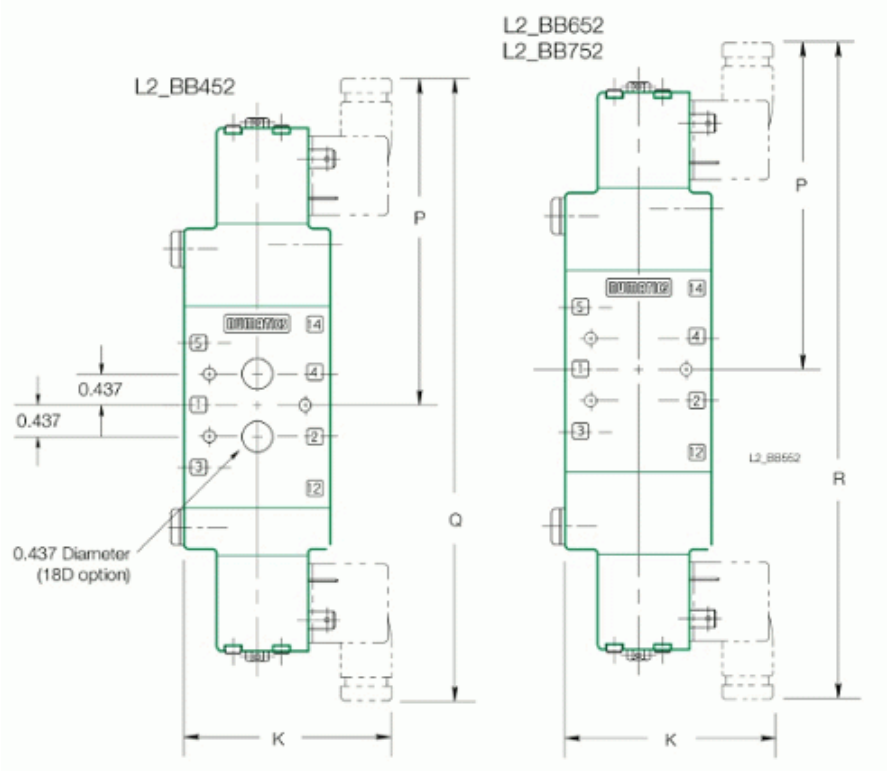


Figura D. 5: *Diagrama de flujo electroválvulas de 4 o 5 vías.*

E. Requisitos para utilizar la plataforma

Previamente a la utilización de la plataforma es conveniente leer y entender esta sección, aquí se indicarán los requisitos técnicos necesarios para su uso. Formas de conexión, arranque de la tarjeta, actualización e instalación de software necesario.

E.1. Primera conexión vía USB, instalación de controladores

Conectamos la Beaglebone al ordenador usando un cable USB A a mini-B, los LEDs empiezan a parpadear. Tras unos 20 segundos, tiempo que tarda en arrancar, los leds se estabilizan e indican:

- USR0: está configurado para parpadear durante el arranque.
- USR1: está configurado para encenderse durante los accesos a la tarjeta microSD.
- USR2: está configurado para encenderse durante la actividad de la CPU.
- USR3: está configurado para encenderse durante los accesos al eMMC.

Aunque en la tarjeta viene instalada una versión de Linux, esta arrancará como unidad flash drive, Figura E. 1, permitiendo el acceso a la documentación y controladores de la tarjeta. Se recomienda hacer una copia de esta información.

Dependiendo del sistema operativo que utilicemos; Mac, Linux o Windows; accedemos a la carpeta BeagleBone Getting Started/Drivers, dentro de esta carpeta seleccionamos el sistema operativo y versión correspondiente y ejecutamos como administradores. Comenzará a instalación de los controladores de la BBB.

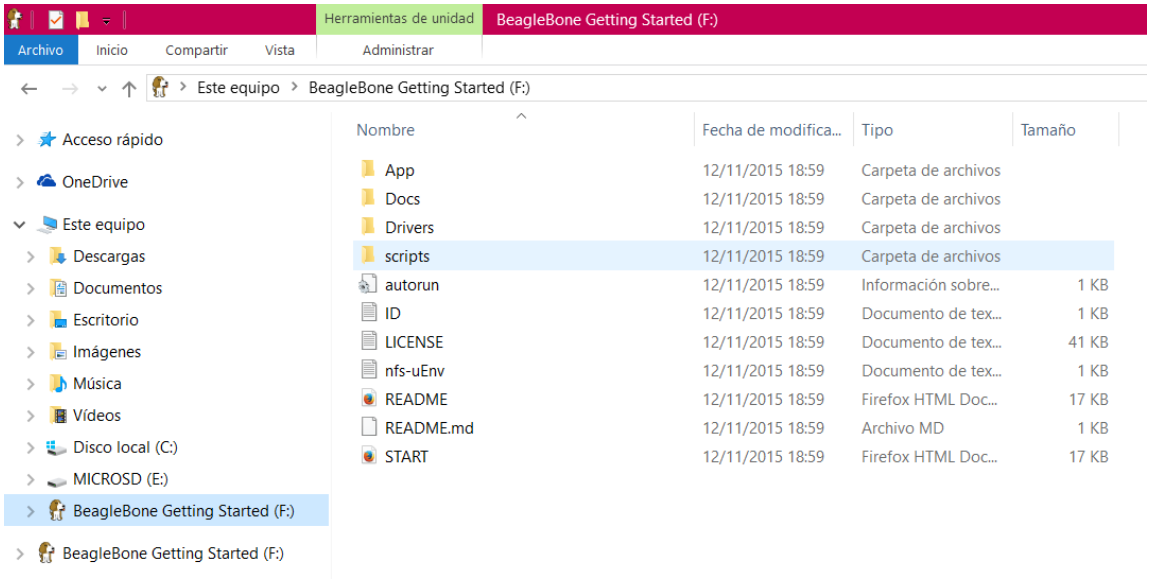


Figura E. 1: *Acceso a la BBB como unidad flash drive.*

Tras la instalación de los drivers ya se puede conectar con la tarjeta vía SSH. Como prueba rápida, en vamos a http://192.168.7.2/ en el navegador de internet (no vale internet explorer). Si conecta, la instalación de los drivers se ha realizado correctamente, ya estamos conectados a la BBB, véase la Figura E. 2..

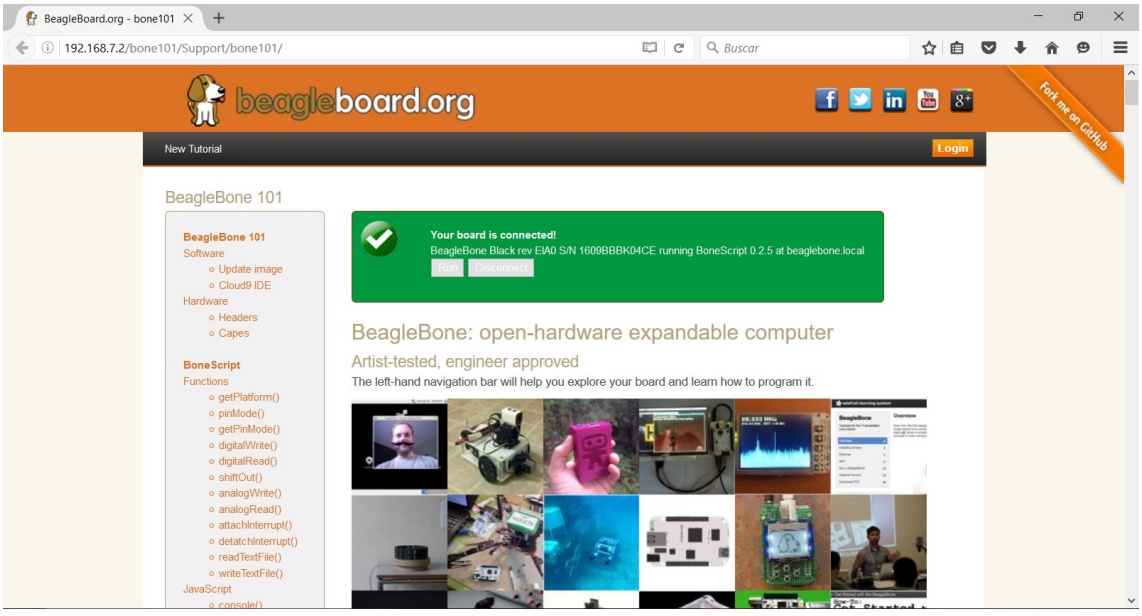


Figura E. 2: *Conexión por Internet a través de USB y un buscador web.*

E.2. Conexión SSH vía USB

En el ordenador, se abre la terminal y se conecta con la BeagleBone. Siguiendo el paso a, b o c dependiendo de tu sistema operativo.

a. Mac: abrir la terminal que se encuentra en */Applications/Utilities/*, y ejecute como administrador:

```
ssh root@192.168.7.2
```

b. Linux: abra la terminal y teclee: `ssh -X root@192.168.7.2`

c. Windows: descargue e instale el programa PuTTY. Una vez instalado ábralo e introduzca con dirección host 192.168.7.2, seleccione SSH como tipo de conexión y presione conectar. Aparecerá una ventana con "login as:", introduzca *root* y presione enter.

La primera vez que nos conectemos a la BBB de esta forma, recibiremos un mensaje advirtiendo que nos estamos conectando a un host desconocido. Se puede omitir este mensaje.

Por defecto, la tarjeta viene configurada sin contraseña de acceso, por lo que cuando por terminal se pregunte por la contraseña bastará con pulsar enter. Una vez conectado se verá la siguiente línea de comandos:

```
root@beaglebone:~#
```

E.2.1. Ventajas y desventajas.

Para este tipo de conexión como se ha indicado sólo es necesario la BBB, un cable USB to miniUSB y acceso a un ordenador con privilegios de administrador. En la siguiente tabla se describen las principales ventajas e inconvenientes la conexión SSH vía USB.

VENTAJAS	DESVENTAJAS
Proporciona una configuración de red estable para principiantes	Está limitado a una única BBB por escritorio
Cuando no tienes acceso o control de una red, sigue proporcionando acceso a internet	La configuración de uso compartido de red puede ser difícil. Es necesario hacer configuración adicional en la BBB.
La energía es suministrada por el propio equipo al que se conecta por USB	El ordenador al que se conecta debe de estar en ejecución con el fin de transmitir datos a internet.

Tabla 22: *Ventajas y Desventajas de la conexión BBB por SSH vía USB.*

En este punto, tenemos la BBB conectada a una red privada, donde la BBB tiene un IP fija 192.168.7.2 y el ordenador la 192.168.7.1. Esto permite la comunicación entre ambos, pero no dota a la BBB de acceso a internet. Algo que va a ser necesario para poder descargar archivos o hacer actualizaciones del sistema en la BBB.

E.2.2. Compartir red con la BBB por conexión SSH vía USB.

Para poder conectar la BBB a internet es necesario compartir el adaptador de red, de forma que el tráfico de la BBB pueda ser enrutado a través del ordenador hacia internet.

A continuación, se explicará el procedimiento para conectar la BBB a internet si se trabaja sobre sistema operativo Linux. En caso de tener MAC o Windows se puede hacer uso del libro [26] de donde se ha sacado la información aquí expuesta. Aunque si eres usuario de alguno de estos dos sistemas operativos, es recomendable que usar una máquina virtual con Linux.

1. Nos conectamos a la BBB por SSH vía USB. Una vez conectados, tecleamos `date` en la terminal y veremos información como la que se ve en la figura de abajo. Como se puede observar en la Figura E. 3, la fecha no se corresponde con la actual.

Figura E. 3: *Fecha y hora de la BBB al conectar por SSH vía USB. Imagen tomada el 02/06/2016.*

A la vista de la imagen queda que la fecha en la BBB no está bien, esto es debido a que, al tratarse de un sistema embebido, a diferencia de un ordenador de sobremesa, no hay reserva de batería para asegurarse recordar de la configuración del BIOS.

2. En la terminal del ordenador de sobremesa (al que se conecta la BBB) escribe `ifconfig` o `ip addr` y pulsa intro. Se desplegará en la pantalla las interfaces de red

conectadas. Busca el adaptador principal (ej eth0) y la conexión con la BBB (ej eth1). En la siguiente Figura E. 4 se muestra el resultado devuelto de la búsqueda.

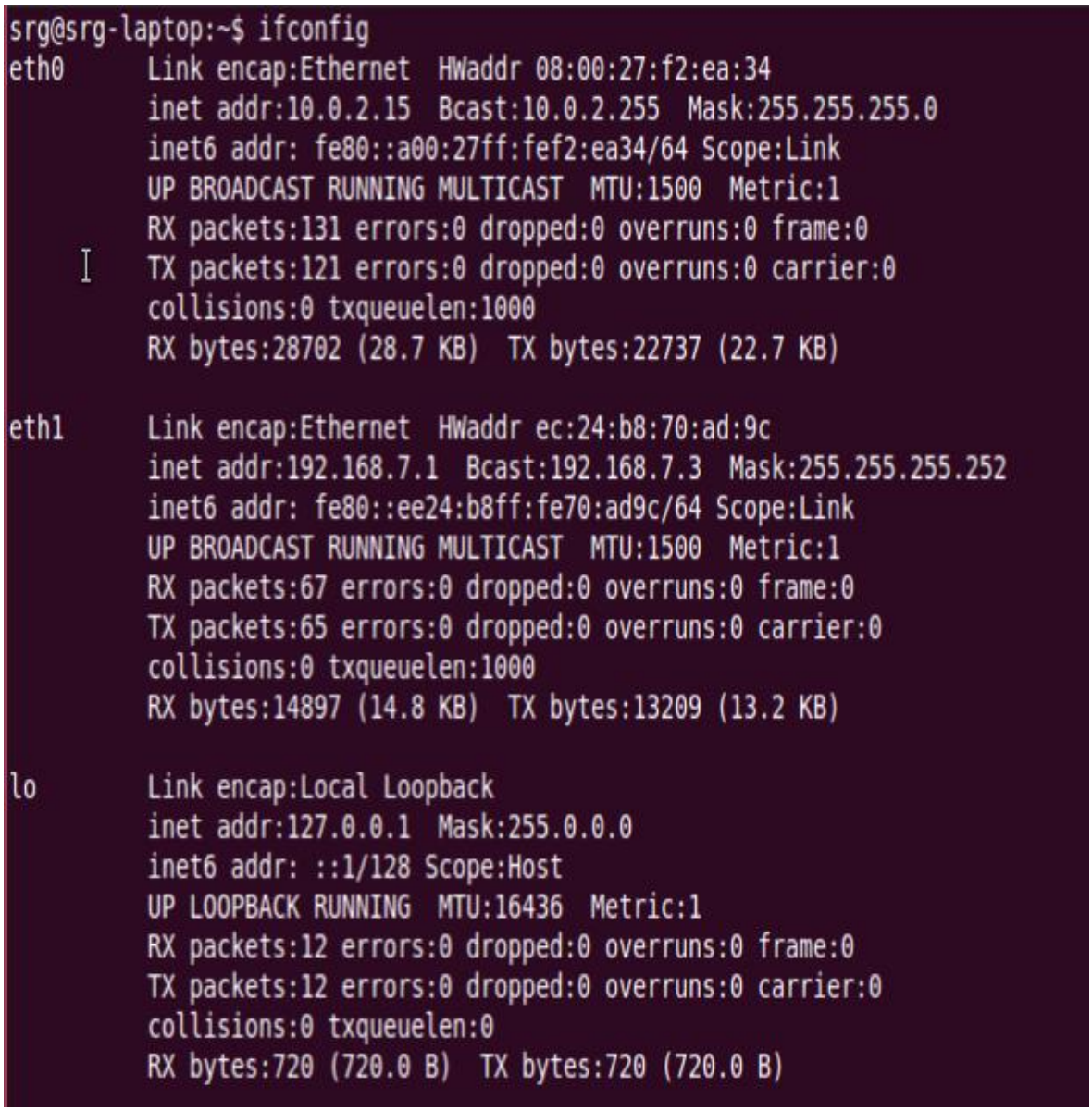


Figura E. 4: *Interfaces de red conectadas. Resultado de ejecutar ifconfig por terminal.*

3. Usar el programa de iptables para configurar las reglas de firewall del kernel de Linux. Logearse como administrador (haciendo sudo su e introduciendo contraseña en la terminal) y ejecutar las siguientes tres líneas de comandos.

```
ifconfig eth1 192.168.7.1
iptables --table nat --append POSTROUTING --out-interface eth0 -j
MASQUERADE
iptables --append FORWARD --in-interface eth1 -j ACCEPT
```

4. Ahora, se ejecuta el siguiente comando para activar el reenvío IP.

```
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

5. Volvemos a la terminal BBB y ejecutamos otras dos líneas de comandos, para indicarle a la tarjeta la dirección IP que debe utilizar como router.

```
route add default gw 192.168.7.1
echo "nameserver 8.8.8.8" > /etc/resolv.conf
```

Con esto ya tenemos la BBB conectada a internet, para comprobarlo podemos realizar un ping a cualquier url. En nuestro caso realizamos: ping www.google.es y obtenemos:

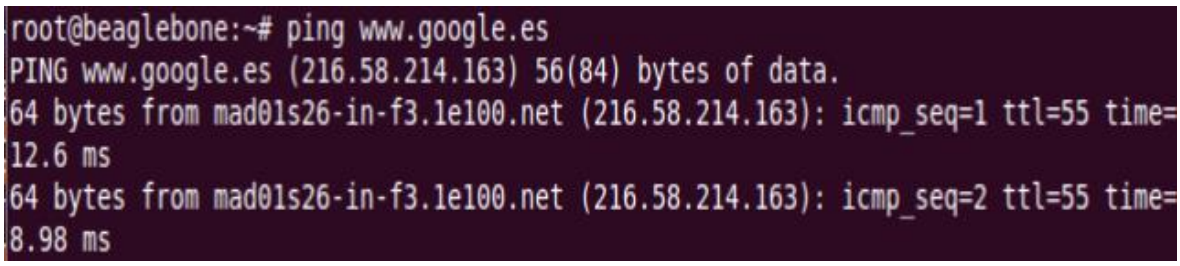


Figura E. 5: *Comprobación de que la BBB conectada por SSH vía USB tiene acceso a internet. Ping a www.google.es*

Se ha solucionado el problema de acceso a internet, pero aún no hemos configurado la fecha Si ejecutamos date de nuevo en la terminal de la BBB, veremos que la fecha ha cambiado, esto es debido a que la BBB ahora la está obteniendo de internet.

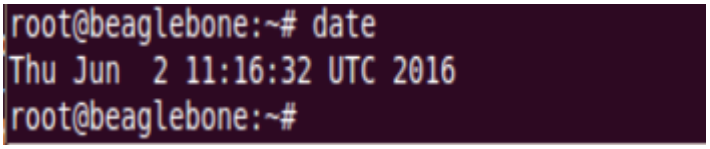


Figura E. 6: *Fecha y hora BBB tras conectarla a internet. Imagen tomada el 02/06/2016 a las 13:16:32.*

Como se aprecia en la Figura E. 6, la fecha ahora es correcta, mientras que la hora no coincide con la hora española. Pero esto es fácil de cambiar haciendo uso del NTP (Network Time Protocol), se trata de un protocolo de red para la sincronización de relojes entre ordenadores. Para fijar la hora a GMT+1 sólo es necesario ejecutar las siguientes dos líneas de comandos:

```
sudo ntpdate pool.ntp.org
export TZ=Europe/Madrid
```

NOTA: es posible que, si es la primera conexión con la BBB o si esta acaba de ser actualizada, el comando `ntpdate` no funcione y se obtenga un diciendo que el comando no se encuentra. Esto es porque aún no está instalado el paquete que contiene este comando en la BBB. Aunque en la sección [E.7] se explicara con mayor exactitud los paquetes instalados en la BBB, si hemos obtenido un error en este punto basta con instalar el comando en la BBB ejecutando las dos siguientes sentencias:

`apt-get update`	Busca las posibles actualizaciones de paquetes disponibles.
`apt-get install ntpdate`	Instala el paquete ntpdate.

Todos los pasos realizados en esta sección serán necesarios realizarlos siempre que se conecte la BBB por SSH vía USB si se quiere que tenga acceso a internet. Es por esto que se desaconseja este tipo de conexión y por ende es recomendable la conexión vía Ethernet explicada en la siguiente sección.

No obstante, como en ocasiones es necesario realizar este tipo de conexión se han realizado dos archivos ejecutables de Linux, uno para la BBB y otro para el ordenador anfitrión, con el fin de facilitar esta configuración. Los ejecutables se pueden encontrar adjuntados y explicados en el Anexo [F].

E.3. Conexión SSH vía Ethernet

Otras veces, será de utilidad conectar la BBB por SSH a través de ethernet. Para la realización de este proyecto esta ha sido la conexión típica, puesto que no se recomienda conectar por USB si se está utilizando una máquina virtual. La mayor parte de este proyecto se desarrolla con el apoyo de un ordenador que tiene instalado Windows 10 y una máquina virtual corriendo Linux.

Pasos para conectar vía SSH sobre Ethernet:

1. Conecta la BBB a la corriente con el trasformador de 5V.

2. Conectar la BBB al router con un cable Ethernet.

3. Acceder al router para saber la dirección local de la beaglebone. Para ello introducir 192.168.1.1 en el buscador web y pulsar enter. Esto llevara a la página de configuración del router, buscar en ella la beaglebone y apuntar su dirección IP. En adelante cuando aparezca beaglebone.local nos referimos a la IP apuntada.

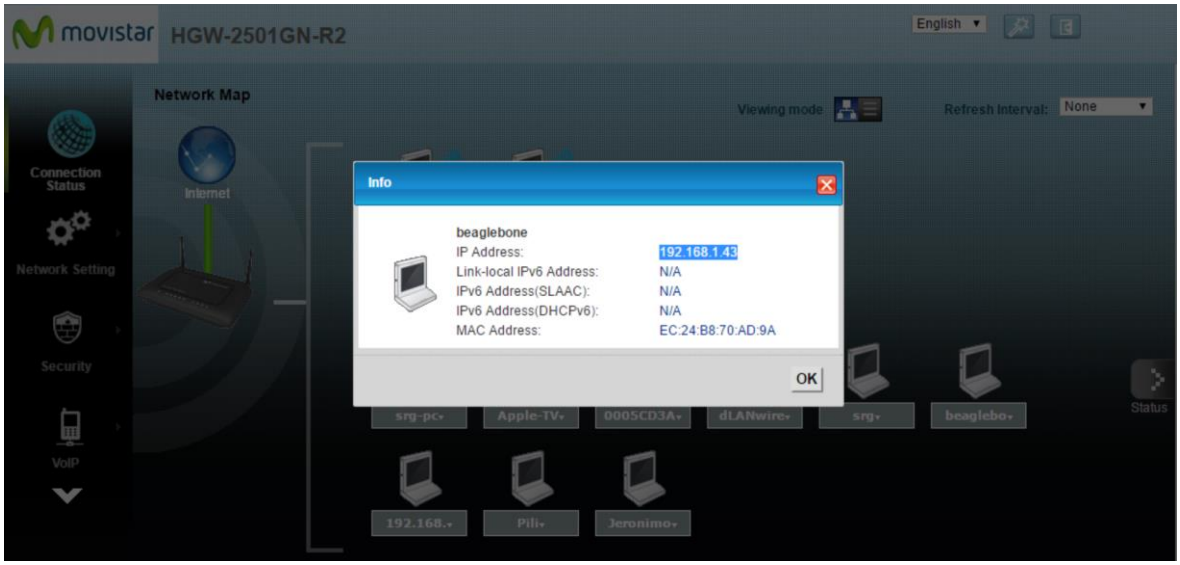


Figura E. 7: *Acceso al router. Dirección IP de la BBB.*

4. Conectar por SSH, dependiendo del sistema operativo:

    a. Mac: abrir la terminal que se encuentra en */Applications/Utilities/*, y ejecute como administrador `ssh root@beaglebone.local`

    b. Linux: abra la terminal y teclee `ssh -X root@beaglebon.local`

    La línea a ejecutar sería: `ssh -X root@192.168.1.43`

    c. Windows: descargue e instale el programa PuTTY. Una vez instalado ábralo e introduzca con dirección host beaglebone.local, seleccione SSH como tipo de conexión y presione conectar. Aparecerá una ventana con "login as:", introduzca `root` y presione enter.

5. Es probable que la primera vez que se conecte de esta forma el cliente SSH de un aviso como que el host es desconocido. Se puede omitir este mensaje.

Por defecto, la tarjeta viene configurada sin contraseña de acceso, por lo que cuando por terminal se pregunte por la contraseña bastará con pulsar enter. Una vez conectado se verá la siguiente línea de comandos: `root@beaglebone:~#`

E.3.1. Ventajas y desventajas

Para este tipo de conexión como se ha indicado sólo es necesario la BBB, el transformador de 5V, un cable de ethernet y acceso tanto a un ordenador con privilegios de administrador como a la red. En la siguiente tabla se describen las principales ventajas e inconvenientes la conexión SSH vía Ethernet.

VENTAJAS	DESVENTAJAS
Ofrece control total sobre la configuración de dirección IP y los ajustes de IP estáticas y dinámicas.	Se necesita tener control de administrador o conocimiento de la infraestructura de la red.
Se pueden conectar muchas BBBs a la misma red (incluyendo dispositivos inalámbricos)	Es necesario un transformador para proporcionar energía a la placa.
La BBB puede estar conectada a internet sin necesidad que un ordenador de sobremesa lo esté a la vez.	La configuración es algo más compleja para principiantes. Si la estructura de red es compleja.

Tabla 23: *Ventajas y desventajas de la conexión BBB por SSH vía Ethernet.*

E.4. Actualización sistema operativo BBB

Llegados a este punto, conviene comprobar que versión de sistema operativo está corriendo en la BBB y en caso de no ser la más actual realizar su actualización. Es conveniente asegurarse primero que las librerías indicadas en la sección [E.8.1] estén actualizadas para la nueva versión del sistema operativo que se desee instalar, en caso contrario es mejor no actualizar pues surgirán problemas con dichas librerías.

Para ver que versión de sistema operativo corre en la tarjeta bastante con ejecutar como administrador la siguiente sentencia en la terminal: `lsb_release -a`

Obteniendo, en nuestro caso, como resultado:

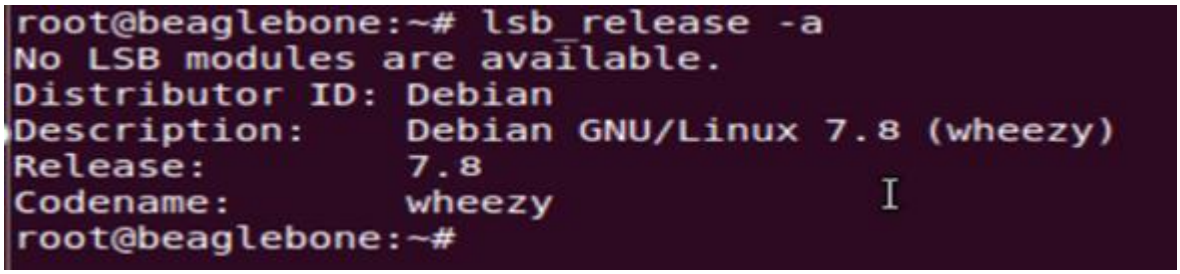
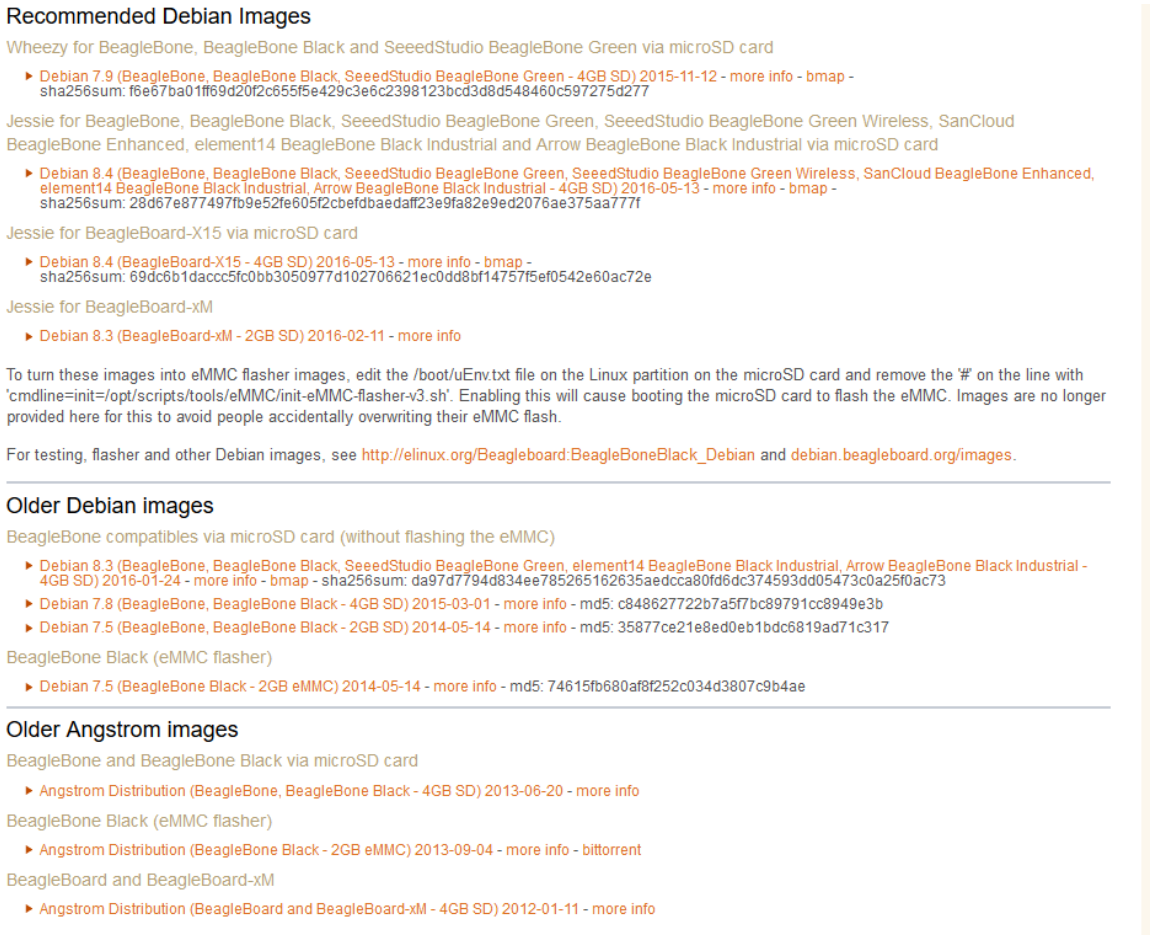


Figura E. 8: *Versión de sistema operativo instalado en la BBB de fábrica.*

En http://beagleboard.org/latest-images se pueden encontrar las últimas versiones operativas para BeagleBone tanto de Debian como de Amstrong. Vamos a dicha página y comprobamos si hay disponible una versión más actual de SO. En nuestro caso, obtenemos lo siguiente:



Recommended Debian Images

Wheezy for BeagleBone, BeagleBone Black and SeeedStudio BeagleBone Green via microSD card

- Debian 7.9 (BeagleBone, BeagleBone Black, SeeedStudio BeagleBone Green - 4GB SD) 2015-11-12 - more info - bmap - sha256sum: f6e67ba01ff69d20f2c655f5e429c3e6c2398123bcd3d8d548460c597275d277

Jessie for BeagleBone, BeagleBone Black, SeeedStudio BeagleBone Green, SeeedStudio BeagleBone Green Wireless, SanCloud BeagleBone Enhanced, element14 BeagleBone Black Industrial and Arrow BeagleBone Black Industrial via microSD card

- Debian 8.4 (BeagleBone, BeagleBone Black, SeeedStudio BeagleBone Green, SeeedStudio BeagleBone Green Wireless, SanCloud BeagleBone Enhanced, element14 BeagleBone Black Industrial, Arrow BeagleBone Black Industrial - 4GB SD) 2016-05-13 - more info - bmap - sha256sum: 28d67e877497fb9e52fe605f2cbefdbaedaff23e9fa82e9ed2076ae375aa777f

Jessie for BeagleBoard-X15 via microSD card

- Debian 8.4 (BeagleBoard-X15 - 4GB SD) 2016-05-13 - more info - bmap - sha256sum: 69dc6b1daccc5fc0bb3050977d102706621ec0dd8bf14757f5ef0542e60ac72e

Jessie for BeagleBoard-xM

- Debian 8.3 (BeagleBoard-xM - 2GB SD) 2016-02-11 - more info

To turn these images into eMMC flasher images, edit the /boot/uEnv.txt file on the Linux partition on the microSD card and remove the '#' on the line with 'cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh'. Enabling this will cause booting the microSD card to flash the eMMC. Images are no longer provided here for this to avoid people accidentally overwriting their eMMC flash.

For testing, flasher and other Debian images, see http://elinux.org/Beagleboard:BeagleBoneBlack_Debian and debian.beagleboard.org/images.

Older Debian images

BeagleBone compatibles via microSD card (without flashing the eMMC)

- Debian 8.3 (BeagleBone, BeagleBone Black, SeeedStudio BeagleBone Green, element14 BeagleBone Black Industrial, Arrow BeagleBone Black Industrial - 4GB SD) 2016-01-24 - more info - bmap - sha256sum: da97d7794d834ee785265162635aedcca80fd6dc374593dd05473c0a25f0ac73
- Debian 7.8 (BeagleBone, BeagleBone Black - 4GB SD) 2015-03-01 - more info - md5: c848627722b7a5f7bc89791cc8949e3b
- Debian 7.5 (BeagleBone, BeagleBone Black - 2GB SD) 2014-05-14 - more info - md5: 35877ce21e8ed0eb1bdc6819ad71c317

BeagleBone Black (eMMC flasher)

- Debian 7.5 (BeagleBone Black - 2GB eMMC) 2014-05-14 - more info - md5: 74615fb680af8f252c034d3807c9b4ae

Older Angstrom images

BeagleBone and BeagleBone Black via microSD card

- Angstrom Distribution (BeagleBone, BeagleBone Black - 4GB SD) 2013-06-20 - more info

BeagleBone Black (eMMC flasher)

- Angstrom Distribution (BeagleBone Black - 2GB eMMC) 2013-09-04 - more info - bittorrent

BeagleBoard and BeagleBoard-xM

- Angstrom Distribution (BeagleBoard and BeagleBoard-xM - 4GB SD) 2012-01-11 - more info

Figura E. 9: *Últimas versiones de SO para BBB disponibles en http://beagleboard.org/latest-images.*

En este caso, como podemos observar Figura E. 9, la versión instalada no es la más actual. Está corriendo Debian 7.8, mientras que el web podemos encontrar ya las versiones 8.3 y 8.4. Es importante fijarse en que el sistema operativo a instalar sea adecuando para la BeagleBone a utilizar. En este caso, las versiones más actuales son muy recientes e informándonos en internet hemos encontrado que las librerías de la sección [E.8.1] no están actualizadas, por lo que nos decantaremos por otra versión.

Para este proyecto, y teniendo en cuenta que usamos una BeagleBone Black Industrial, y lo indicado en el párrafo anterior, instalaremos Debian 7.9 Wheezy que es compatible con esta.

Descargamos la imagen software que deseemos. Esta tiene una extensión tipo .img.xz. imagen comprimida sector por sector de la microSD. Por lo que serán necesarios programas para su descompresión y grabado en la tarjeta.

- Programa descompresor de imagen *.img* es el 7-zip que se puede descargar de http://www.7-zip.org/download.html

- Programa para instalar la imagen en la tarjeta es Win32 Disk Imager. Se puede descargar en https://sourceforge.net/projects/win32diskimager/

Tras descargar e instalar los programas, procedemos a realizar la actualización:

1. Usando 7-zip descomprimimos el archivo de imagen software descargado.

2. Conecta la tarjeta microSD al ordenador.

3. Utilizando Win32 Disk Imager. Selecciona el archivo de imagen deseado, y con la casilla MD5 Hash marcada, pulsar el botón *Write.*

4. Esperar hasta que se realice la escritura en la tarjeta y extraerla con seguridad del ordenador.

5. Inserta la microSD en la BBB. Encenderla y conectarse a ella vía SSH. Se recomienda usar Linux.

6. Acceder a la carpeta /media/rootfs/boot/ y abrir el archivo de nombre uEnv.txt. Si usas Linux basta con ejecutar la siguiente sentencia: sudo nano /media/rootfs/boot/uEnv.txt

7. Se abrirá el documento uEnv.txt. Buscar dentro del mismo las siguientes líneas:

```
##enable BBB: eMMC Flasher:
#cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh
```

8. Descomentar la segunda línea, borrar la almohadilla, guardar el documento y apagar la BBB.

9. Con la microSD dentro de la BBB, arrancarla manteniendo el pulsado el botón USER/BOOT, ya sea por USB o con el trasformador de 5V. Esto hará que la imagen grabada en la tarjeta se vuelque a la eMMC de la BBB. Este proceso puede llevar unos 45 minutos, se comenzará a flashear la Beaglebone Black copiando todos los archivos de la microSD Card, en este proceso los leds comenzarán a parpadear uno por uno de forma secuencial de izquierda a derecha y al acabar el proceso se apagará la Beaglebone Black (los cuatro USRx LEDS se apagan).

10. Extraer la microSD, para evitar que se repita el proceso de flasheo y aplicar energía de nuevo. La BeagleBone ya está actualizada y lista para usar.

Para comprobar la instalación podemos volver a ejecutar la sentencia lsb_release -a, en el caso de este proyecto, podemos observar en la Figura E. 10 que ahora en la BBB corre Debian 7.9.

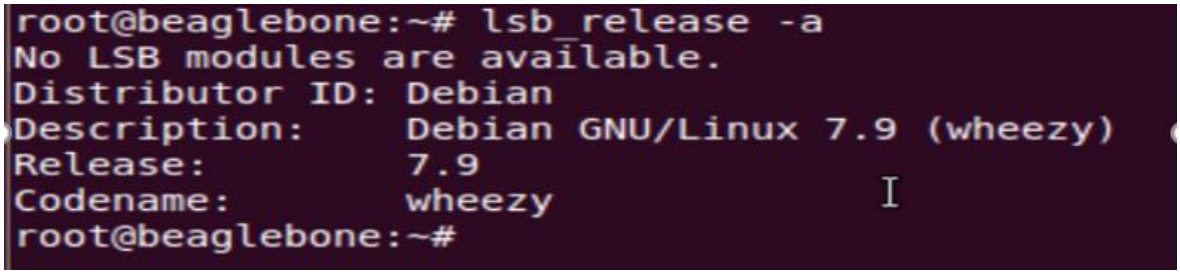


Figura E. 10: *Versión del sistema operativo de la BBB tras la actualización.*

E.5. Apagado de la BBB

Aunque pueda parecer algo trivial, es necesario explicar el correcto procedimiento para el apagado del sistema embebido, ya que, un apagado incorrecto puede dañar gravemente el sistema de archivos o provocar un aumento en el tiempo de arranque.

A continuación, unos puntos importantes para apagar, reiniciar y arrancar la BBB de forma adecuada:

- Escribiendo en la terminal `shutdown -h now`. Se apagar la tarjeta cerrando los procesos o programas abiertos. Esta sentencia permite también apagar al paso de 5 minutos, por ejemplo: `shutdown -h +5`

- Escribiendo en la terminal `reboot` se reinicia adecuadamente la tarjeta.

- Se puede pulsar el botón de apagado/encendido (power button) una vez para un apagado correcto de la placa.

- Manteniendo el botón de apagado/encendido por unos ocho segundos se realiza un apagado forzoso de la BBB. Este método debe evitarse a no ser que la tarjeta esté bloqueada y no sea posible un apagado suave.

- En modo apagado, con el conector de corriente o el USB conectado, presionar el botón de encendido/apagado arranque la tarjeta.

Para una mayor duración de la BBB es conveniente no arrancar la tarjeta desconectando y conectando la toma de corriente, es mejor utilizar el botón de encendido.

E.6. Introducción a Linux

Una vez realizada la primera toma de contacto con la BBB, realizada su actualización y entendidas las distintas formas de conexión que permite. Procedemos a la configuración de

la misma, instalación de programas, librerías y complementos necesarios para el proyecto. Así como una breve introducción a Linux y sus comandos más usados por Shell.

E.6.1. Comandos básicos de Linux

Linux es un sistema operativo de software libre que ofrece la posibilidad de ejecutar cualquier rutina o programa por medio de comandos de Shell, sin necesidad de trabajar con ventanas como se hace en Windows. Para un usuario nuevo en Linux, esta sección le proporcionará suficientes habilidades para salir adelante, se detallarán comandos y referencias con ejemplos para su mejor comprensión.

COMANDO	DESCRIPCIÓN
more /etc/issue	Devuelve la distribución de Linux que se está usando
ps -p $$	Devuelve la versión de shell que se está usando
whoami	Devuelve el usuario con el que se ha iniciado sesión
uptime	Devuelve durante cuánto tiempo lleva corriendo el sistema.
top	Listado de procesos y programas en ejecución

Tabla 24: *Comandos habituales para la terminal Linux.*

Los comandos básicos a la hora de moverse y manipular un archivo en Linux se muestran en la siguiente tabla. Cuando se usa un usuario sin poderes de administrador en Debian o Ubuntu habitualmente es necesario escribir la palabra sudo antes de ciertos comandos que deben de ser ejecutados como administrador.

NOMBRE	COMANDO	OPCIONES E INFORMACIÓN	EJEMPLO
Listar archivos	**ls**	-a (muestra todo, incluidos archivos ocultos) -l (muestra formato largo) -R (da un listado recursivo) -r (da una lista invertida) -t (ordenado por fecha de modificación) -S (muestra el tamaño del fichero)	ls -al
Directorio actual	**pwd**	muestra el directorio de trabajo. -P (muestra la localización física)	pwd -P
Cambiar directorio	**cd**	cd ~/ (lleva directorio home) cd / (lleva al directorio root) cd .. (lleva eleva un directorio)	cd /home/root cd /
Crear directorio	**mkdir**	crea un directorio con nombre X	mkdir test

Borrar un archivo o directorio	**rm**	-r (borrado recursivo) -d (borrado de directorios vacíos)	rm bad.txt rm -r test
Copiar archivo o directorio	**cp**	-r (de forma recursiva) -u (copia solo si la fuente es nueva)	cp a.txt b.txt cp -r test testa
Mover un archivo o directorio	**mv**	-i (pregunta antes de sobrescribir)	mv a.txt c.txt mv test testb
Crear un archivo en blanco	**touch**	Crea un archivo en blanco o actualiza la fecha de modificación de uno existente	touch d.txt
Ver contenido de un archivo	**more**	Usar la barra espaciadora para pasar pagina	more d.txt

Tabla 25: *Comandos básicos para trabajar por terminal con ficheros en Linux.*

Estos son los primeros comandos con los que conviene familiarizarse, aunque más adelante se detallarán más, es apropiado hacer una parada y hablar de los atajos de teclado que funcionan con la terminal de Linux.

ATAJO	DESCRIPCION
Flecha arriba	Muestra el ultimo comando que se ejecutó, y los anteriores si se sigue pulsando la tecla.
Tabulador	Función de autocompletado de nombres de archivos, directorios o ejecutables.
Crtl+A	Te lleva al inicio de la línea que estas escribiendo
Crtl+E	Te lleva al final de la línea que estas escribiendo
Crtl+U	Borra desde el cursor hasta el inicio de la línea
Crtl+L	Borra la pantalla
Crtl+C	Mata el proceso que este corriendo
Crtl+Z	Envía un proceso a background, con fg se saca de segundo plano.

Tabla 26: *Atajos de teclado en la terminal Linux.*

E.6.2. Sistema de gestión de paquetes Linux

Se conoce como sistema de gestión o administrador de paquetes a la colección de herramientas para automatizar el proceso de instalación, actualización y eliminación de paquetes de software. Linux incorpora este sistema.

En estos administradores, el software se distribuye en forma de paquetes, frecuentemente encapsulado en un solo fichero. No contienen sólo el software, sino que, por lo general,

también incluyen el nombre completo, una descripción de su funcionalidad, el número de versión, el nombre del distribuidor y una lista de otros paquetes requeridos para el correcto funcionamiento del software.

Dependiendo de la versión de Linux nos encontramos con diferentes sistemas de gestión de paquetes. Angstrom usa OPKG (Open PacKaGe), Ubuntu y Debian usan APT (Advanced Packaging Tool) sobre DPKG (Debian Package Management System) y Arch Linux usa Pacman. Cada uno de estos usa sintaxis diferente, pero su funcionamiento es muy similar.

En la realización del presente proyecto se ha utilizado una distribución Debian, pero dado que en futuras versiones de la plataforma puede ser necesario actualizar la BBB con alguna otra distribución. En el siguiente cuadro se expondrán las principales sentencias de los administradores OPKG y APT.

COMANDO	ANGSTROM	DEBIAN/UBUNTU
Instalar paquete	**opkg install geany**	**sudo apt-get install geany**
Actualizar paquete	**opkg update**	**sudo apt-get update**
Comprobar si esta un paquete instalado	**opkg list-installed | grep geany**	**dpkg-query -l | grep geany**
Mas informacion del paquete	**opkg info geany**	**apt-cache show geany** **apt-cache policy geany**
Obtener ayuda	**opkg**	**apt-get help**
Descargar un paquete al diretorio actual	**opkg download geany**	**sudo apt-get download geany**
Eliminar un paquete	**opkg remove geany**	**sudo apt-get remove geany**
limpiar los paquetes viejos	No hay sentencia para ello	**sudo apt-get clean**

Tabla 27: *Comandos de gestión de paquetes para terminal Linux.*

E.7. Paquetes

Tras el primer inicio y actualización de la BBB nos encontramos con un sistema operativo que puede tener paquetes desactualizados o puede no tener instalados paquetes que serán necesarios para elaboración de este proyecto. En esta sección se indicarán los paquetes que han sido necesarios instalar, así como, una descripción de su uso y los comandos para su instalación (por si en un trabajo futuro fuera necesario reinstalar los paquetes).

Lo primero a realizar es una actualización de los paquetes ya instalados en el sistema, hay dos comandos importantes en este punto:

- **apt-get update**: actualiza la lista de paquetes disponibles y sus versiones, pero no instala o actualiza ningún paquete. Esta lista la coge de los servidores con repositorios que están definidos en sources.list.

- **apt-get upgrade**: una vez el comando update ha descargado la lista de software disponible y la versión en la que se encuentra, podemos actualizar dichos paquetes usando este comando. Instalará las nuevas versiones respetando la configuración del software cuando sea posible.

Estos comandos es recomendable ejecutarlos siempre que se instale un nuevo paquete, pues buscará su versión más actual y lo actualizará.

E.7.1. Instalación de paquetes fundamentales

7.2.3.1. Geany.

Se trata de un pequeño y ligero editor basado en Scintilla con características básicas de entorno de desarrollo integrado (IDE). Cuyas características principales son:

- Sirve para programar en más de 50 lenguajes como C, Java, Pascal, HTML, C, C++, Python y muchos más.
- Tiene función de autocompletado.
- Se le puede instalar plugins para añadirle funcionalidades.
- Permite plegar el código por secciones, y así tener una vista general de todo el texto.
- Es un entorno ligero y fácil de usar.
- Colorea el código en función del lenguaje que se utilice.
- Compilación y ejecución del código.
- Permite búsqueda de textos concretos en el código.
- Numera las líneas del documento.

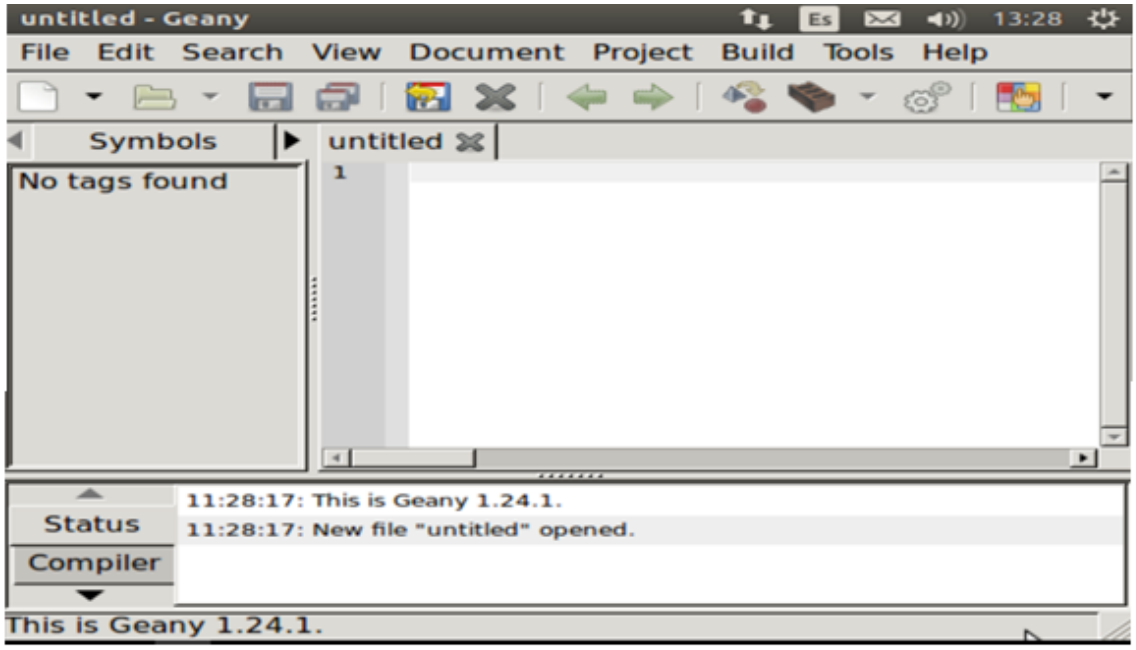


Figura E. 11: *Entorno de programación Geany.*

Para instalar el programa en Debian nos dirigimos a la terminal y lanzamos el siguiente comando:

```
sudo apt-get install geany geany-common
```

Ya tenemos instalado Geany, para utilizarlo basta con escribir en la terminal geany para que se arranque el programa, su aspecto se muestra en la Figura E. 11.

7.2.3.2. Pcmanfm

Se trata de un administrador de ficheros y archivos. Es un gestor de archivos de estándar LXDE, desarrollado por Hong Jen. Permite moverse por las carpetas de fichero por medio de ventanas.

Las características principales son:

- Miniaturas de imágenes.
- Íconos de escritorio.
- Marcadores de internet.
- Gestión de escritorio.
- Utilidad de búsqueda de ficheros.

- Navegación por pestañas.
- Administración de volúmenes incorporada (montar/desmontar/expulsar).
- Permite arrastrar y soltar.
- Los ficheros se pueden arrastrar entre pestañas.
- Multilingüe.
- Asociación de ficheros (aplicación por defecto).
- Proporciona vista de iconos, vista compacta, vista de lista detallada y vista en miniatura.
- En modo *root* permite la manipulación de archivos protegidos del sistema, previa autentificación como superusuario.

Para instalar el paquete basta con abrir la terminal y ejecutar el siguiente comando:

```
sudo apt-get install pcmanfm
```

Ya está instalado el administrador de archivos, para ejecutarlo basta con introducir en la terminal pcmanfm, la apariencia del gestor se muestra en la Figura E. 12:

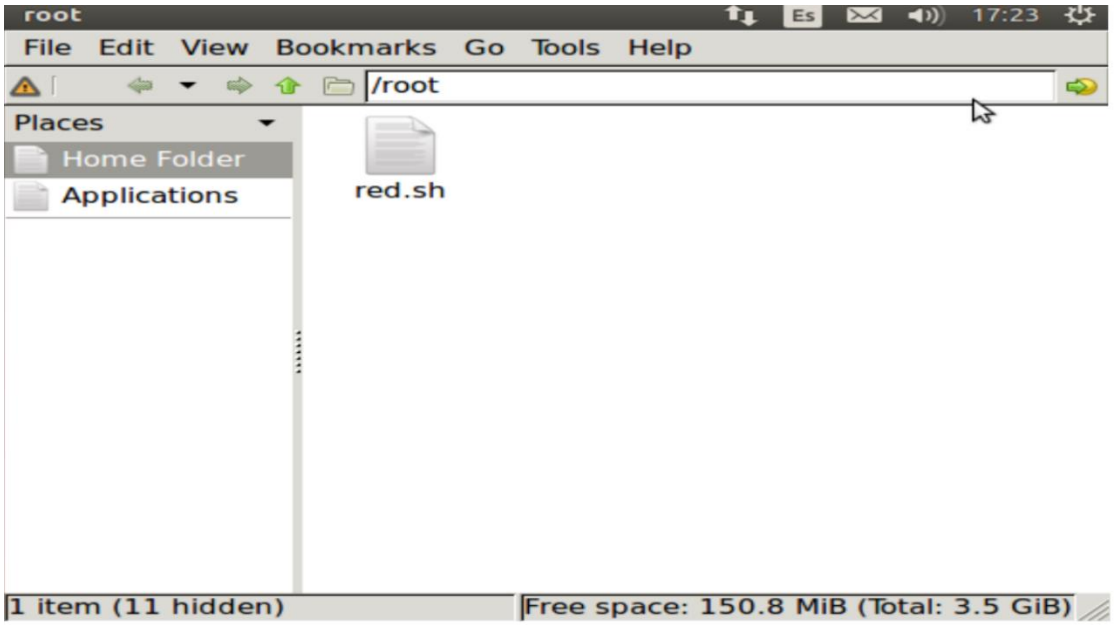


Figura E. 12: *interfaz gráfica del gestor de archivos pcmanfm.*

7.2.3.3. Gnuplot

Se trata de un programa muy flexible que permite generar gráficas 2D y 3D. Sus principales virtudes son un acabado de muy alta calidad y su facilidad de uso.

Se trata de un programa de software libre en el sentido en que las fuentes están disponibles (y además son gratuitas), permite su copia y modificación. Pero no se permite distribuir versiones modificadas del mismo, estas sólo se pueden distribuir en forma de parches.

Las principales características de Gnuplot son:

- Realiza representaciones bidimensionales con distintos estilos (puntos, líneas, barras…).
- Realiza representaciones tridimensionales (contorno y superficie).
- Facilidades para etiquetar las gráficas, ejes y puntos representados (títulos y etiquetas).
- Permite realizar cálculos con enteros, decimales y complejos.
- Funciona en distintos SO y permite obtener gráficos en casi cualquier formato.
- Posee un conjunto de funciones predefinidas y permite al usuario definir las suyas propias.
- Permite trabajo interactivo o en modo comando (shell).

Este programa ya viene instalado por defecto en Debian.

E.8. Python

Es un lenguaje de programación poderoso, intuitivo y fácil de comprender. Su filosofía hace hincapié en una sintaxis que favorezca un código legible. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo en la programación orientada a objetos. Su elegante sintaxis junto con un tipado dinámico hacen de este un lenguaje ideal para scriting y desarrollo rápido de aplicaciones, Además es multiplataforma.

Posee licencia de código abierto, con lo que se ha favorecido el desarrollo de módulos libres de Python de terceros que facilitan la interactuación con ciertos dispositivos. El intérprete de Python y la extensa biblioteca estándar están disponibles en forma binaria y de código fuente para las principales plataformas en el sitio web de Python, http://www.python.org/.

Python permite escribir programas más compactos y legibles que sus homólogos en C, C++ o Java. Esto es debido a que los tipos de datos de alto nivel permiten expresar operaciones complejas en una sola instrucción, a que la agrupación de instrucciones se hace por sangría en vez de por llaves y a que no es necesario declarar variables ni argumentos. Además, ofrece más chequeo de error que C, y siendo un lenguaje de muy alto nivel, tiene tipos de datos de alto nivel incorporados como arreglos de tamaño flexible y diccionarios.

Este interprete permite extenderse fácilmente con nuevas funcionalidades y tipos de datos implementados en C o C++ (u otros lenguajes accesibles desde C).

Se trata de un lenguaje de programación multiparadigma. Esto significa que, en vez de forzar al programador a adoptar un estilo de programación determinado, permite varios estilos. Por ejemplo, programación orientada a objetos, programación imperativa y programación funcional. Además, se pueden utilizar otros paradigmas con el uso de extensiones.

Permite separar el programa a ejecutar por módulos que pueden ser reusados por otros programas en Python. Viene con una gran colección de módulos estándar que puedes usar como base, estos módulos proveen cosas como entrada/salida a archivos, llamadas al sistema sockets, etc. Esto será muy útil para el desarrollo de nuestro programa, pues se utilizarán módulos de terceros con el fin de facilitar la comunicación con dispositivos externos a la BBB.

Por todo lo antes expuesto es que se ha elegido Python como lenguaje de programación para el proyecto. Actualmente, existen disponibles dos versiones la 2.7.3 y la 3.5.1. Dado que por defecto en el Debian 7.9 instalado en la BBB viene instalada la versión 2.7.3, y ya que además se trata de la versión más antigua (hay más librerías y módulos disponibles), será la versión con la que se trabaje.

Antes de comenzar a utilizar Python es conveniente leerse el tutorial que está disponible en su página web [28].

E.8.1. Librerías y módulos Python

Como ya se ha indicado, Python posee una gran cantidad de módulos y librerías, a continuación, se indicarán y describirán las librerías o módulos necesarios para el correcto funcionamiento de la plataforma de odorante.

E.8.1.1. Adafruit_BBIO library

Se trata de una librería (libre) creada por el equipo de desarrollo de Adafruit, que permite el acceso a los pines de entrada/salida de la BeagleBone utilizando Python. Nos permitirá configurar los GPIO en sus distintos modos de operación, así como activarlos o desactivarlos en forma de salida o entrada. En esta sección no se profundizará en el uso de la librería sino en su instalación, para comprender la funcionalidad de la misma ver la sección [5.3].

Para instalar la librería basta con seguir los siguientes pasos:

1. Con la BBB con conexión a internet, por terminal ejecutamos el siguiente comando para asegurarnos que la BBB tiene la fecha y hora correctas:

```
sudo ntpdate pool.ntp.org
```

2. Ahora se instalan las dependencias:

```
sudo apt-get update
sudo apt-get install build-essential python-dev python-setuptools python-
pip python-smbus -y
```

3. Se instala la librería:

```
sudo pip install Adafruit_BBIO
```

Con esto la librería ya está instalada y lista para utilizarse.

E.8.1.2. Adafruit_Python_DHT

Se trata de un módulo (libre) creado por el equipo de desarrollo de Adafruit que permite leer de forma rápida el valor de temperatura y humedad del sensor DHT22. Este módulo contiene código en C para comunicarse con el sensor DHT, ya que requieren tiempo que la lectura se haga muy rápidamente. Este código C luego se envuelve en una librería de Python para su fácil utilización.

Para instalar la librería basta con seguir los siguientes pasos:

1. Con la BBB con conexión a internet, por terminal ejecutamos el siguiente comando para asegurarnos que la BBB tiene la fecha y hora correctas:

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
cd Adafruit_Python_DHT
```

2. Se instalan las dependencias:

```
sudo apt-get update
sudo apt-get install build-essential python-dev python-openssl
```

3. Se instala la librería:

```
sudo python setup.py install
```

Con esto la librería ya está instalada y lista para utilizarse.

E.8.1.3. Librería SciPy

Es una biblioteca open source de herramientas y algoritmos matemáticos para Python. Contiene módulos para optimización, álgebra lineal, procesamiento de señales, FFT, funciones especiales, etc.

Para instalar la librería basta con ejecutar por terminal la siguiente línea cuando la BBB está conectada a internet:

```
sudo apt-get install python-scipy
```

E.8.1.4. Librería NumPy

Es un módulo open source de Python que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel.

Para instalar la librería basta con ejecutar por terminal la siguiente línea cuando la BBB está conectada a internet:

```
sudo apt-get install python-numpy
```

F. Ejecutables de conexión SSH por vía USB

Para facilitar la conexión a internet cuando la BBB está conectada por USB se han creado dos ejecutables, uno que se ejecutará en el PC anfitrión y uno segundo que se ejecutará en la BeagleBone.

El primero se ejecuta en el ordenador anfitrión y se ha llamado *conexión.sh*. Para su realización se ejecutó la siguiente sentencia en terminal para crear el archivo:

```
sudo nano conexión.sh
```

Y en el archivo se han introducido las siguientes sentencias:

```
#!/bin/bash
# -*- ENCODING: UTF-8 -*-

#COMPARTIR INTERNET CON BBB

#se configuran con iptables las reglas de firewall del kernel.
ifconfig eth2 192.168.7.1
iptables –table nat –append POSTROUTING –out-interface eth0 -j MASQUERADE
iptables –append FORWARD –in-interface eth2 -j ACCEPT

#se active el reenvio IP
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Se guarda el archivo, pero aún no está listo para utilizarse. Antes hay que darle permisos de ejecución, para ello lanzamos la siguiente sentencia:

```
chmod +x conexión.sh
```

El segundo se ejecuta en la BeagleBone y se le ha llamado *red.sh*. Para su creación se ejecutó la siguiente sentencia en terminal para crear el fichero:

```
sudo nano red.sh
```

Y en el archivo se han introducido las siguientes líneas de comandos:

```
#!/bin/bash
# -*- ENCODING: UTF-8 -*-

#COMPARTIR INTERNET POR USB

#se fija en la BBB la dirección IP a utilizar como router
route add default gw 192.168.7.1
echo “nameserver 8.8.8.8” > /etct/resolv.conf
```

```
#se sincroniza el reloj
ntpdate pool,ntp.org
export TZ=Europe/Madrid
```

Al igual que el primer archivo hace falta darle permisos de ejecución, lanzando la siguiente sentencia:

```
chmod +x red.sh
```

Es importante ejecutar los dos archivos en orden y en modo superusuario (*root*) o sino no funcionará. Primero se lanza en el PC anfitrión el ejecutable *conexión.sh* y después en BBB se lanza el ejecutable *red.sh*. Para lanzar un ejecutable basta con escribir en terminal:

`./conexión.sh` y `./red.sh`

G. Código calculo resistencia de carga del Sensor

```
% TGS2600 10k~90k in air (1k~90k in others)

Vin=5
y_min=0;
y_max=Vin;
Rdiv0=10000
Rdiv=Rdiv0;
x_min=0;
x_max=90*1000;
Rs=(x_min:.1:x_max);


figure()
subplot(4,3,1)
plot(Rs,(1000./(Rs+1000))*Vin)
xlabel('Rs'); ylabel('Vout');title(['Rdiv=1000 Ohms , Vin= ' num2str(Vin)
' Vol'])

for i=2:10
subplot(4,3,i)
plot(Rs,(Rdiv./(Rs+Rdiv))*Vin)
xlabel('Rs'); ylabel('Vout');title(['Rdiv=' num2str(Rdiv) ' Ohms , Vin= '
num2str(Vin) ' Vol']);
xlim([x_min x_max]);ylim([y_min y_max]);
rdiv(i)=Rdiv;
Rdiv=Rdiv+Rdiv0;
end
```

H. Códigos de Ejecución

H.1. Código experimentación pura sin modulación

```
# -*- encoding: utf-8 -*- # Especificamos que nuestro archivo .py esta
codificado en UTF-8
#!/usr/bin/python


"""
      Plataforma de experimentacion

    SERGIO DE LA CRUZ GUTIERREZ

    ALGORITMO PURO SIN MODULACION
    Parametros:

            Succion motor (50-100)
            Tiempo conmutacion electrovalvula (en segundos)
            Tiempo de experimentacion (en minutos)

      Sentencia de ejecucion: tgs2600PURO.py succion switch tiempo

"""


import Adafruit_BBIO.ADC as ADC
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.PWM as PWM
import Adafruit_DHT
import sys
import os
import time
import math
import random
import numpy
from scipy import stats
from datetime import datetime, date
import thread

##    ASIGNACION DE PUERTOS.

electrovalve1 = 'P8_10'       #Electrovalvula 1 (METANOL)
electrovalve2 = 'P8_12'       #Electrovalvula 2 (ETANOL)
electrovalve3 = 'P8_14'       #Electrovalvula 3 (BUTANOL)
electrovalve4 = 'P8_16'       #Electrovalvula 4 (AIRE)

motorPin = 'P9_21'            #Motor de succion

sensorPin2600 = 'P9_38'       #Sensor TGS2600 (Puerto AIN_3)

sensorTemp22 =  Adafruit_DHT.DHT22
Temp22 = 'P8_11'        #Pin de lectura de temperatura/Humedad con DHT22
(Puerto GPIO_45)
```

```
##    DECLARACION DE SALIDAS DEL SISTEMA.

GPIO.setup("P8_10",GPIO.OUT)
GPIO.setup("P8_12",GPIO.OUT)
GPIO.setup("P8_14",GPIO.OUT)
GPIO.setup("P8_16",GPIO.OUT)


##    DECLARACION DE VARIABLES DEL SISTEMA.

# Regresion.

NM=10       #Numero lecturas ADC
T=0.1             #Las NM lecturas se hacen en T segundos
tsub=T/NM;        #Subdivisiones de tiempo para las lecturas ADC
SLEEP=1           #Periodo de estabilizacion cuando se pone una nueva
temperatura
SAMPLESINICIO=10        #Capturas iniciales

x=[]        #Almacena el numero de muestra
concentTGS2600=[]       #Almacena las muestras de odorante


# Sensor TGS2600.

Rl_2600=440 #Omh
Vc=5 #V

# Measure temperatura y humedad
SLEEP_tyh = 59


nameFile = "puroTGS2600.txt"  #Fichero de informacion con parámetros y
lecturas de la experimentacion
nameFile_data = "puroTGS2600.dat"   #Fichero de salida de datos
nameFile_tyh = "TyH_TGS2600.data"   #Fichero de informacion de
temperatura y humedad dela experimentacion
fileString = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime())+'_'
+nameFile
fileString_data =time.strftime("%a%d%b%Y-%HH%MM%SS",
time.localtime())+'_' +nameFile_data
fileString_tyh = time.strftime("%a%d%b%Y-%HH%MM%SS",
time.localtime())+'_' +nameFile_tyh
ahora=datetime.now()
ruta = "/media/MICROSD/PURO/"+str(ahora.month)+"/"+str(ahora.day)+"/"
      #Ruta de salida
if not os.path.exists(ruta): os.makedirs(ruta) #Si la ruta no existe, se
crea
ruta_fichero = ruta.strip() + str(fileString)
ruta_fichero_data = ruta.strip() + str(fileString_data)
ruta_fichero_tyh = ruta.strip() + str(fileString_tyh)


## Creacion de los ficheros de datos y Temperatura/Humedad del
experimento.

f = open(ruta_fichero, "w+")
g=open(ruta_fichero_data, "w+")
```

```
h = open(ruta_fichero_tyh, "w+")

##    FUNCION de creacion del fichero de información del experimento.
Sensor TGS2600 (modo escritura) y cabecera

def file_TGS2600(vec_open_valve,succion,switch,tiempo):

      f.write
('////////////////////////////////////////////////////////////////////\n
')
      f.write ('\n\nPlataforma de experimentacion: \n\nSERGIO DE LA CRUZ
GUTIERREZ\n\n')
      f.write ('Sensor TGS2600\n')
      f.write ('Algoritmo PURO SIN MODULACION\n')
      date=time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime())
      f.write ('Fecha y hora de inicio: ' + str(date) + '\n')
      f.write ('Ruta del fichero: ' + str(ruta) + '\n')
      f.write ('Nombre del fichero: ' + str(nameFile) + '\n')
      f.write ('Nombre del fichero de datos: ' + str(nameFile_data) +
'\n')
      f.write ('Nombre del fichero de TyH: ' + str(nameFile_tyh) + '\n')
      f.write ('Electrovalvula (1-METANOL, 2-ETANOL, 3-BUTANOL, 4-AIRE )
\n')
      f.write ('Conmutacion entre electrovalvulas: '
+str(vec_open_valve)+ '\n')
      f.write ('\n')
      f.write ('Parametros para la experimetacion, se introducen como
argumentos \n')
      f.write ('Succion motor (50-100)    >>> '+ str(succion)+ '%\n')
      f.write ('Tiempo de conmutacion de electrovalcula (en segundos)
      >>> '+str(switch)+'\n\n')
      f.write ('Duracion del esperimento: ' +str(tiempo)+ ' minutos\n')
      f.write
('////////////////////////////////////////////////////////////////////\n
\n')
      f.write ('  CAPTURA DE DATOS:\n\n')


#     Apertura electrovalvula

def apertura(electrovalvula):
      if electrovalvula == 1:
            GPIO.output(electrovalve1, GPIO.HIGH)
            GPIO.output(electrovalve2, GPIO.LOW)
            GPIO.output(electrovalve3, GPIO.LOW)
            GPIO.output(electrovalve4, GPIO.LOW)
      elif electrovalvula == 2:
            GPIO.output(electrovalve1, GPIO.LOW)
            GPIO.output(electrovalve2, GPIO.HIGH)
            GPIO.output(electrovalve3, GPIO.LOW)
            GPIO.output(electrovalve4, GPIO.LOW)
      elif electrovalvula == 3:
            GPIO.output(electrovalve1, GPIO.LOW)
            GPIO.output(electrovalve2, GPIO.LOW)
            GPIO.output(electrovalve3, GPIO.HIGH)
            GPIO.output(electrovalve4, GPIO.LOW)
      elif electrovalvula == 4:
            GPIO.output(electrovalve1, GPIO.LOW)
            GPIO.output(electrovalve2, GPIO.LOW)
            GPIO.output(electrovalve3, GPIO.LOW)
```

```
            GPIO.output(electrovalve4, GPIO.HIGH)


#     Arranque motor

def motor_start(succion):
      PWM.start(motorPin,succion)


#     Funcion de medida de temperatura y humedad

def measure_tyh(tiempo):

      j=0
      tiempo=round(tiempo)
      time.sleep(9)
      #Lectura humedad y temperatura
      while j<tiempo:
            tick_HT = time.time()
            humidity, temperature = Adafruit_DHT.read_retry(sensorTemp22,
Temp22)
            instante = datetime.now()
            tack_HT=time.time()
            t_HT=tack_HT-tick_HT


            #Cuanto duerme en funcion de lo que tarde en H y T
            if t_HT > SLEEP_tyh:
                  print "Tiempo medicion H y T > SLEEP:", t_HT
            else:
                  print "Tiempo medicion H y T:", t_HT


                  if humidity is not None  and temperature is not None:
                        print'\nSensor DHT22: ' +str(sensorTemp22)
                        print'>>> '+str(instante)+'   Temp = '
+str(temperature)+ '    Humidity = ' +str(humidity)+'\n'
                        h = open(ruta_fichero_tyh, "a")
                        wline_h=str(instante)+' '+str(temperature)+'
      '+str(humidity)+'\n'
                        h.writelines(wline_h)
                        h.flush()

                  else :
                        print 'Failed to get reading, Try again!'
                        h = open(ruta_fichero_tyh, "a")
                        wline_h=str(instante)+' '+str(temperature)+'
      '+str(humidity)+'\n'
                        h.writelines(wline_h)
                        h.flush()
            tack=time.time()
            time.sleep(SLEEP_tyh-(tack-tick_HT))
            j+=1



#     Samplesinicio Puro TGS2600

def ini_puro_TGS2600(count):
```

```
	random.seed()

	#Lectura nariz

	value=0
	queda=0
	residuo=0
	ADC.read_raw(sensorPin2600) #la primera medida es erronea por el
bug
	for i in range(NM):
		value += ADC.read_raw(sensorPin2600)
		r=random.uniform(0, tsub)
		time.sleep(r)
		residuo += (tsub-r)
	time.sleep(residuo);
	valueTGS2600=value/NM
	instante_captura=datetime.now()

	x.append(count)
	concentTGS2600.append(valueTGS2600)

	#Se calcula el valor de la resistencia interna del sensor
	RsTGS2600=((Vc*Rl_2600)/(valueTGS2600/1000.))-Rl_2600

	#Escritura por pantalla de la lectura
	print 'Muestra_ini['+str(count)+']\n>> Valor:
'+str(valueTGS2600)+'mV >> Rs: '+str(RsTGS2600)+'          >>
'+str(instante_captura)+'\n'

	#Se escriben los datos en el fichero
	f = open(ruta_fichero, "a")
	g = open(ruta_fichero_data, "a")
	wline_g=str(count)+'    '+str(valueTGS2600)+'   '+str(RsTGS2600)+'
		'+str(instante_captura)+'\n'
	wline_f='Muestra_ini['+str(count)+']
	Valor:'+str(valueTGS2600)+'mV -- Rs: '+str(RsTGS2600)+'
	>>'+str(instante_captura)+'\n'
	g.writelines(wline_g)
	g.flush()
	f.writelines(wline_f)
	f.flush()



#	Puro TGS2600

def puro_TGS2600(count):

	value=0
	queda=0
	residuo=0

	ADC.read_raw(sensorPin2600) #la primera medida es erronea por el
bug
	for i in range(NM):
		value += ADC.read_raw(sensorPin2600)
		r=random.uniform(0, tsub)
		time.sleep(r)
		residuo += (tsub-r)
	time.sleep(residuo); #sleep+(tsub*NM)+T+residuo=1seg
```

```
      valueTGS2600=value/NM
      instante_captura=datetime.now()

      x.append(count)
      concentTGS2600.append(valueTGS2600)

      #Se calcula el valor de la resistencia interna del sensor
      RsTGS2600=((Vc*Rl_2600)/(valueTGS2600/1000.))-Rl_2600

      #Escritura por pantalla de la lectura
      print 'Muestra['+str(count)+']\n>> Valor: '+str(valueTGS2600)+'mV
      >> Rs: '+str(RsTGS2600)+'     >> '+str(instante_captura)+'\n'

      #Se escriben los datos en el fichero
      f = open(ruta_fichero, "a")
      g = open(ruta_fichero_data, "a")
      wline_g=str(count)+'    '+str(valueTGS2600)+'   '+str(RsTGS2600)+'
            '+str(instante_captura)+'\n'
      wline_f='Muestra['+str(count)+']    Valor:'+str(valueTGS2600)+'mV --
Rs: '+str(RsTGS2600)+'  >>'+str(instante_captura)+'\n'
      g.writelines(wline_g)
      g.flush()
      f.writelines(wline_f)
      f.flush()


#     Cierre

def cierre():

      PWM.stop(motorPin)
      PWM.cleanup()
      thread.exit()
      fecha_fin=datetime.now()
      print('\nEXPERIMENTO FINALIZADO CON EXITO')
      print 'Experimento terminado: '+str(fecha_fin)
      f = open(ruta_fichero, "a")
      f.write('\nEXPERIMENTO FINALIZADO CON EXITO\n')
      f.write('Fecha y hora de fin de experimentacion:     '
+str(fecha_fin))
      f.flush()
      f.close()
      g.close()
      h.close()


def main():

      if(len(sys.argv)<4):
            print '\n\nPARAMETROS INCORRECTOS. \n' \
            'Los parametros deben ser:\n' \
            'Succion motor (50-100) \n' \
            'Tiempo conmutacion electrovalvula (en segundos) \n' \
            'Tiempo de experimentacion (en minutos)\n'
            return 0

      print '\nSensor: TGS2600    Pin ADC: ' +sensorPin2600
      print 'Algoritmo: PURO SIN MODULACION'
      print 'Ruta Fichero: ' +ruta_fichero
      print 'Ruta Fichero de datos: '+ruta_fichero_data
```

```
      #SUCCION MOTOR:   50-100%

      succion = float(sys.argv[1])
      if succion > 100 or succion < 50:
            print '\n\nPARAMETRO MOTOR INCORRECTO.\n' \
            'Los parametros deben ser:\n' \
            'Succion motor (50-100) \n' \
            'Tiempo conmutacion electrovalvula (en segundos) \n' \
            'Tiempo de experimentacion (en minutos)\n'
            return 0
      else :
            print 'Succion motor al ' +str(succion)+ '%    motor Pin PWM
: ' +motorPin

      #TIEMPO CONMUTACION ELECTROVALVULA

      switch = float(sys.argv[2])
      if switch <1:
            print '\n\nPARAMETRO TIEMPO CONMUTACION ELECTROVALVULA
INCORRECTO.\n' \
            'Los parametros deben ser:\n' \
            'Succion motor (50-100) \n' \
            'Tiempo conmutacion electrovalvula (en segundos) \n' \
            'Tiempo de experimentacion (en minutos)\n'
            return 0
      else :
            print 'Tiempo conmutacion electrovalvula: ' +str(switch)+'
      minutos'

      #DURACION DEL EXPERIMENTO

      tiempo = float(sys.argv[3])
      if tiempo < (switch/60):
            print '\n\nPARAMETRO TIEMPO EXPERIMENTACION INCORRECTO.\n' \
            'Los parametros deben ser:\n' \
            'Succion motor (50-100) \n' \
            'Tiempo conmutacion electrovalvula (en segundos) \n' \
            'Tiempo de experimentacion (en minutos)\n'
            return 0
      else :
            print 'Tiempo de experimentación: ' +str(tiempo)+ '
minutos\n'

      raw_input("Si es correcto presiona enter, sino ctr-c\n") #Se
comprueba por pantalla que los parametros son correctos

      #Se calcula de forma aleatoria la conmutacion de electrovalvulas
      conmuta=((tiempo*60)/switch)
      vec_open_valve = numpy.random.randint(1,4,conmuta)
      print 'Conmutacion entre electrovalvulas:  ' +str(vec_open_valve)+
'\n'

      #Llamada al thread de medida de temperatura y humedad
      thread.start_new_thread(measure_tyh, (tiempo,))

      #Se realiza el setup de los puertos
      motor_start(succion)
      ADC.setup()
```

```
	#Calentamiento del sensor, se toman SAMPLESINICIO medidas antes de
comenzar la experimentacion
	count=1
	while count<=SAMPLESINICIO:
		tick_puro=time.time()
		ini_puro_TGS2600(count)
		count+=1
		tack_puro=time.time()
		time.sleep(SLEEP-(tack_puro-tick_puro))

	print '\n\nComienza la experimentacion\n'
	f.write('\n\nComienza la experimentacion\n')
	f.flush()

	#Comienza la experimentacion

	i=0

	while count<=(tiempo*60) + SAMPLESINICIO :

		electrovalvula = vec_open_valve[i]

		if electrovalvula == 1:
			print 'ELECTROVALVULA: '+str(electrovalvula)+' METANOL
\n'
		elif electrovalvula == 2:
			print 'ELECTROVALVULA: '+str(electrovalvula)+' ETANOL
\n'
		elif electrovalvula == 3:
			print 'ELECTROVALVULA: '+str(electrovalvula)+' BUTANOL
\n'

		apertura(electrovalvula)

		j=0
		while j<switch:
			tick_puro=time.time()
			puro_TGS2600(count)
			count+=1
			j+=1
			tack_puro=time.time()
			time.sleep(SLEEP-(tack_puro-tick_puro))

		i+=1


	#Terminar
	cierre()
	return 0


if __name__ == '__main__':
	try:
		main()
	except KeyboardInterrupt:
		print'Interrupted'
		cierre()
		sys.exit(0)
```

H.2. Código modulación Martinelli

```
# -*- encoding: utf-8 -*- # Especificamos que nuestro archivo .py esta
codificado en UTF-8
#!/usr/bin/python

"""
      Plataforma de experimentacion

    SERGIO DE LA CRUZ GUTIERREZ

    ALGORITMO MARTINELLI
    Parametros:
            Succion motor (50-100)
            Duracion de la experimentacion (en muestras)
            Tiempo de apertura de cada válvula (en segundos)



      Sentencia de ejecucion: tgs2600martinelli.py succion muestras
switch

"""

##    Declaracion de librerias a utilizar.

import Adafruit_BBIO.ADC as ADC
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.PWM as PWM
import Adafruit_DHT
import sys
import os
import time
import math
import random
import numpy
from scipy import stats
from datetime import datetime, date
import thread


##    ASIGNACION DE PUERTOS.

electrovalve1 = 'P8_10'       #Electrovalvula 1 (METANOL)
electrovalve2 = 'P8_12'       #Electrovalvula 2 (ETANOL)
electrovalve3 = 'P8_14'       #Electrovalvula 3 (BUTANOL)
electrovalve4 = 'P8_16'       #Electrovalvula 4 (AIRE)

motorPin = 'P9_21'            #Motor de succion

sensorPin555 = 'P9_12'        #Sensor TGS2600 tras conversion con 555
(Puerto GPIO_60)
heatPin2600 = 'P9_14'         #Calentamiento sensor TGS2600 (Puerto
GPIO_50)

sensorTemp22 =  Adafruit_DHT.DHT22
```

```
Temp22 = 'P8_11'        #Pin de lectura de temperatura/Humedad con DHT22
(Puerto GPIO_45)


##    DECLARACION DE ENTRADAS/SALIDAS DEL SISTEMA.

GPIO.setup("P8_10",GPIO.OUT)
GPIO.setup("P8_12",GPIO.OUT)
GPIO.setup("P8_14",GPIO.OUT)
GPIO.setup("P8_16",GPIO.OUT)
GPIO.setup("P9_12",GPIO.IN)         #Salida 555


##    DECLARACION DE VARIABLES DEL SISTEMA.

# Martinelli.

SAMPLESINICIO=1   #Capturas iniciales
heat2600=100      #Temperatura de calentamiento del sensor

x=[]        #Almacena el numero de muestra
up=[]       #Almacena el tiempo de los pulsos en estado alto
down=[]           #Almacena el tiempo de los pulsos en estado bajo
tempTGS2600=[]          #Almacena las temperaturas de calentamiento del
sensor
duracion=[]       #Almacena la duracion de cada pulso
duracion_k=[]     #Almacena la duracion de k pulso

# Sensor TGS2600. Lectura ADC

concentTGS2600=[]       #Almacena las muestras de odorante (lectura ADC)
SLEEP_ADC = 1
NM=10       #Numero lecturas ADC
T=0.1       #Las NM lecturas se hacen en T segundos
tsub=T/NM;        #Subdivisiones de tiempo para las lecturas ADC
Rl_2600=27000 #Ohm
Vc=5 #V

# Measure temperatura y humedad

SLEEP_tyh = 59


##    NOMBRE Y RUTA DE LOS FICHEROS DE SALIDA.

nameFile = "Martinelli_TGS2600.txt" #Fichero de informacion con
parámetros y lecturas de la experimentacion
nameFile_data1 = "Martinelli_data1.dat"  #Fichero de salida de datos,
duracion de los pulsos Martinelli
nameFile_data2 = 'Martinelli_data2.dat'  #Fichero da salida de datos,
duracion de los k pulsos
nameFile_tyh = "TyH_TGS2600.dat"    #Fichero de informacion de
temperatura y humedad dela experimentacion
fileString = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime())+'_'
+nameFile
fileString_data1 = time.strftime("%a%d%b%Y-%HH%MM%SS",
time.localtime())+'_' +nameFile_data1
fileString_data2 = time.strftime("%a%d%b%Y-%HH%MM%SS",
time.localtime())+'_' +nameFile_data2
```

```
fileString_tyh = time.strftime("%a%d%b%Y-%HH%MM%SS",
time.localtime())+'_' +nameFile_tyh
ahora=datetime.now()
ruta =
"/media/MICROSD/FRECUENCIA/MARTINELLI/"+str(ahora.month)+"/"+str(ahora.da
y)+"/"            #Ruta de salida
if not os.path.exists(ruta): os.makedirs(ruta) #Si la ruta no existe, se
crea
ruta_fichero = ruta.strip() + str(fileString)
ruta_fichero_data1 = ruta.strip() + str(fileString_data1)
ruta_fichero_data2 = ruta.strip() + str(fileString_data2)
ruta_fichero_tyh = ruta.strip() + str(fileString_tyh)


## Creacion de ficheros del experimento.

g=open(ruta_fichero_data1, "w+")    #fichero de datos del experimento,
Duracion de cada pulso.
k=open(ruta_fichero_data2, "w+")    #fichero que guarda la duracion de
los 16 pulsos
h = open(ruta_fichero_tyh, "w+")    #fichero que guarda la temperatura y
humedad

##    FUNCION de creacion del fichero salida sensor TGS2600 (modo
escritura) y cabecera.

def file_TGS2600(vec_open_valve,succion,heat2600,SAMPLES):

      f = open(ruta_fichero, "w+")
      f.write
('////////////////////////////////////////////////////////////////////\n
')
      f.write ('\n\nPlataforma de experimentacion: \n\nSERGIO DE LA CRUZ
GUTIERREZ\n\n')
      f.write ('Sensor TGS2600\n')
      f.write ('Algoritmo MARTINELLI\n')
      date=time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime())
      f.write ('Fecha y hora de inicio: ' + str(date) + '\n')
      f.write ('Ruta del fichero: ' + str(ruta) + '\n\n')
      f.write ('Nombre del fichero: ' + str(fileString) + '\n')
      f.write ('Nombre del fichero de datos 1: ' + str(fileString_data1)
+ '\n')
      f.write ('Nombre del fichero de datos 2: ' + str(fileString_data2)
+ '\n\n')
      f.write ('Nombre del fichero de TyH: ' + str(nameFile_tyh) + '\n')
      f.write ('Electrovalvula (1-METANOL, 2-ETANOL, 3-BUTANOL, 4-AIRE )
\n\n')
      f.write ('Conmutacion entre electrovalvulas: '
+str(vec_open_valve)+ '\n')
      f.write ('\n')
      f.write ('Parametros para la experimetacion, se introducen como
argumentos \n')
      f.write ('Succion motor (50-100)    >>> '+ str(succion)+ '%\n')
      f.write ('Duracion del experimento: ' +str(SAMPLES)+ ' muestras\n')
      f.write
('////////////////////////////////////////////////////////////////////\n
\n')
      f.write ('  CAPTURA DE DATOS:\n\n')
```

```
##    DECLARACION DE FUNCIONES DEL SISTEMA.

# Apertura electrovalvula.

def apertura(electrovalvula):
      if electrovalvula == 1:
            GPIO.output(electrovalve1, GPIO.HIGH)
            GPIO.output(electrovalve2, GPIO.LOW)
            GPIO.output(electrovalve3, GPIO.LOW)
            GPIO.output(electrovalve4, GPIO.LOW)
      elif electrovalvula == 2:
            GPIO.output(electrovalve1, GPIO.LOW)
            GPIO.output(electrovalve2, GPIO.HIGH)
            GPIO.output(electrovalve3, GPIO.LOW)
            GPIO.output(electrovalve4, GPIO.LOW)
      elif electrovalvula == 3:
            GPIO.output(electrovalve1, GPIO.LOW)
            GPIO.output(electrovalve2, GPIO.LOW)
            GPIO.output(electrovalve3, GPIO.HIGH)
            GPIO.output(electrovalve4, GPIO.LOW)
      elif electrovalvula == 4:
            GPIO.output(electrovalve1, GPIO.LOW)
            GPIO.output(electrovalve2, GPIO.LOW)
            GPIO.output(electrovalve3, GPIO.LOW)
            GPIO.output(electrovalve4, GPIO.HIGH)


# Arranque motor.

def motor_start(succion):
      PWM.start(motorPin,succion)


#     Funcion de medida de temperatura y humedad

def measure_tyh(tiempo):

      j=0
      time.sleep(9)
      #Lectura humedad y temperatura
      while (tiempo == True):
            tick_HT = time.time()
            humidity, temperature = Adafruit_DHT.read_retry(sensorTemp22,
Temp22)
            instante = datetime.now()
            tack_HT=time.time()
            t_HT=tack_HT-tick_HT


            #Cuanto duerme en funcion de lo que tarde en H y T
            if t_HT > SLEEP_tyh:
                  print "Tiempo medicion H y T > SLEEP:", t_HT
            else:
                  print "Tiempo medicion H y T:", t_HT


                  if humidity is not None  and temperature is not None:
                        print'\nSensor DHT22: ' +str(sensorTemp22)
```

```
                        print'>>> '+str(instante)+'   Temp = '
+str(temperature)+ '    Humidity = ' +str(humidity)+'\n'
                        h = open(ruta_fichero_tyh, "a")
                        wline_h=str(instante)+' '+str(temperature)+'
      '+str(humidity)+'\n'
                        h.writelines(wline_h)
                        h.flush()

                  else :
                        print 'Failed to get reading, Try again!'
                        h = open(ruta_fichero_tyh, "a")
                        wline_h=str(instante)+' '+str(temperature)+'
      '+str(humidity)+'\n'
                        h.writelines(wline_h)
                        h.flush()

            tack=time.time()
            time.sleep(SLEEP_tyh-(tack-tick_HT))
            j+=1



#     Samplesinicio Martinelli 2600

def samplesinicio_martinelli_TGS2600():

      count = 0
      value_down = GPIO.wait_for_edge(sensorPin555, GPIO.FALLING)
      #Espera pulso de caida
      #print '0\n'
      ini_pulso = time.time()
      while count <= 7 :

            temperature_TGS2600=60
            PWM.set_duty_cycle(heatPin2600, temperature_TGS2600)

            value_up = GPIO.wait_for_edge(sensorPin555, GPIO.RISING)
            #Espera pulso de subida
            ini_up = time.time()
            time_down = ini_up - ini_pulso           #Duracion del pulso
en estado bajo
            #print '1\n'

            value_down = GPIO.wait_for_edge(sensorPin555, GPIO.FALLING)
            #Espera pulso de bajada
            fin_pulso = time.time()
            instante_captura=datetime.now()
            time_up = fin_pulso - ini_up       #Duracion del pulso en
estado alto
            time_pulso = fin_pulso-ini_pulso   #Duracion del pulso
completo
            ini_pulso = time.time()
            #print '0\n'

            x.append(count)
            up.append(time_up)
            down.append(time_down)
            duracion.append(time_pulso)

            #Escritura por pantalla de la lectura
```

```
            print 'Muestra Martinelli_ini['+str(count)+']\n>>
t_up='+str(time_up)+'   >> t_down: '+str(time_down)+ '      >> Durancion
pulso: '+str(time_pulso)
            print '>> Heat: '+str(temperature_TGS2600)+'   >>
'+str(instante_captura)+'\n'

            #Se escriben los datos en el fichero
            f=open(ruta_fichero, "a")
            g = open(ruta_fichero_data1, "a")
            wline_g=str(count)+'    '+str(time_up)+'  '+str(time_down)+'
      '+str(time_pulso)+'     '+str(temperature_TGS2600)+'
      '+str(instante_captura)+'\n'
            wline_f='Muestra_ini['+str(count)+']
      t_up='+str(time_up)+'   t_down: '+str(time_down)+ '    Durancion
pulso: '+str(time_pulso)+'    Heat: '+str(temperature_TGS2600)+'>>
'+str(instante_captura)+'\n'
            g.writelines(wline_g)
            g.flush()
            f.writelines(wline_f)
            f.flush()

            count+=1

      while count > 7 and count <= 15 :

            temperature_TGS2600=100
            PWM.set_duty_cycle(heatPin2600, temperature_TGS2600)

            value_up = GPIO.wait_for_edge(sensorPin555, GPIO.RISING)
      #Espera pulso de subida
            ini_up = time.time()
            time_down = ini_up - ini_pulso           #Duracion del pulso
en estado bajo
            #print '1\n'

            value_down = GPIO.wait_for_edge(sensorPin555, GPIO.FALLING)
            #Espera pulso de bajada
            fin_pulso = time.time()
            instante_captura=datetime.now()
            time_up = fin_pulso - ini_up       #Duracion del pulso en
estado alto
            time_pulso = fin_pulso-ini_pulso   #Duracion del pulso
completo
            ini_pulso = time.time()
            #print '0\n'

            x.append(count)
            up.append(time_up)
            down.append(time_down)
            duracion.append(time_pulso)

            #Escritura por pantalla de la lectura
            print 'Muestra Martinelli_ini['+str(count)+']\n>>
t_up='+str(time_up)+'   >> t_down: '+str(time_down)+ '      >> Durancion
pulso: '+str(time_pulso)
            print '>> Heat: '+str(temperature_TGS2600)+'   >>
'+str(instante_captura)+'\n'

            #Se escriben los datos en el fichero
            f=open(ruta_fichero, "a")
```

```
            g = open(ruta_fichero_data1, "a")
            wline_g=str(count)+'    '+str(time_up)+'  '+str(time_down)+'
      '+str(time_pulso)+'     '+str(temperature_TGS2600)+'
      '+str(instante_captura)+'\n'
            wline_f='Muestra_ini['+str(count)+']
      t_up='+str(time_up)+'   t_down: '+str(time_down)+ '    Durancion
pulso: '+str(time_pulso)+'    Heat: '+str(temperature_TGS2600)+'>>
'+str(instante_captura)+'\n'
            g.writelines(wline_g)
            g.flush()
            f.writelines(wline_f)
            f.flush()

            count+=1


#     Martinelli TGS2600

def martinelli_TGS2600():

      count = 0
      value_down = GPIO.wait_for_edge(sensorPin555, GPIO.FALLING)
      #Espera pulso de bajada
      #print '0\n'
      ini_pulso = time.time()
      while count <= 7 :

            temperature_TGS2600=60
            PWM.set_duty_cycle(heatPin2600, temperature_TGS2600)

            value_up = GPIO.wait_for_edge(sensorPin555, GPIO.RISING)
      #Espera pulso de subida
            ini_up = time.time()
            time_down = ini_up - ini_pulso           #Duracion del pulso
en estado  bajo
            #print '1\n'

            value_down = GPIO.wait_for_edge(sensorPin555, GPIO.FALLING)
            #Espera pulso de bajada
            fin_pulso = time.time()
            instante_captura=datetime.now()
            time_up = fin_pulso - ini_up       #Duracion del pulso en
estado alto
            time_pulso = fin_pulso-ini_pulso   #Duracion del pulso
completo
            ini_pulso = time.time()
            #print '0\n'

            x.append(count)
            up.append(time_up)
            down.append(time_down)
            duracion.append(time_pulso)

            #Escritura por pantalla de la lectura
            print 'Muestra Martinelli['+str(count)+']\n>>
t_up='+str(time_up)+'   >> t_down: '+str(time_down)+ '      >> Durancion
pulso: '+str(time_pulso)
            print '>> Heat: '+str(temperature_TGS2600)+'   >>
'+str(instante_captura)+'\n'
```

```
            #Se escriben los datos en el fichero
            f=open(ruta_fichero, "a")
            g = open(ruta_fichero_data1, "a")
            wline_g=str(count)+'    '+str(time_up)+'  '+str(time_down)+'
      '+str(time_pulso)+'     '+str(temperature_TGS2600)+'
      '+str(instante_captura)+'\n'
            wline_f='Muestra['+str(count)+']    t_up='+str(time_up)+'
      t_down: '+str(time_down)+ '    Durancion pulso: '+str(time_pulso)+'
      Heat: '+str(temperature_TGS2600)+'>> '+str(instante_captura)+'\n'
            g.writelines(wline_g)
            g.flush()
            f.writelines(wline_f)
            f.flush()

            count+=1

      while count > 7 and count <= 15 :

            temperature_TGS2600=100
            PWM.set_duty_cycle(heatPin2600, temperature_TGS2600)

            value_up = GPIO.wait_for_edge(sensorPin555, GPIO.RISING)
      #Espera pulso de subida
            ini_up = time.time()
            time_down = ini_up - ini_pulso           #Duracion del pulso
en estado bajo
            #print '1\n'

            value_down = GPIO.wait_for_edge(sensorPin555, GPIO.FALLING)
            #Espera pulso de bajada
            fin_pulso = time.time()
            instante_captura=datetime.now()
            time_up = fin_pulso - ini_up       #Duracion del pulso de
subida
            time_pulso = fin_pulso-ini_pulso   #Duracion del pulso
completo
            ini_pulso = time.time()
            #print '0\n'

            x.append(count)
            up.append(time_up)
            down.append(time_down)
            duracion.append(time_pulso)

            #Escritura por pantalla de la lectura
            print 'Muestra Martinelli['+str(count)+']\n>>
t_up='+str(time_up)+'   >> t_down: '+str(time_down)+ '     >> Durancion
pulso: '+str(time_pulso)
            print '>> Heat: '+str(temperature_TGS2600)+'   >>
'+str(instante_captura)+'\n'

            #Se escriben los datos en el fichero
            f=open(ruta_fichero, "a")
            g = open(ruta_fichero_data1, "a")
            wline_g=str(count)+'    '+str(time_up)+'  '+str(time_down)+'
      '+str(time_pulso)+'     '+str(temperature_TGS2600)+'
      '+str(instante_captura)+'\n'
            wline_f='Muestra['+str(count)+']    t_up='+str(time_up)+'
      t_down: '+str(time_down)+ '    Durancion pulso: '+str(time_pulso)+'
      Heat: '+str(temperature_TGS2600)+'>> '+str(instante_captura)+'\n'
```

```
            g.writelines(wline_g)
            g.flush()
            f.writelines(wline_f)
            f.flush()

            count+=1


#     Cierre

def cierre():

      PWM.stop(heatPin2600)
      PWM.stop(motorPin)
      PWM.cleanup()
      thread.exit()
      fecha_fin=datetime.now()
      print('\nEXPERIMENTO FINALIZADO CON EXITO')
      print 'Experimento terminado: '+str(fecha_fin)
      f = open(ruta_fichero, "a")
      f.write('\n\XPERIMENTO FINALIZADO CON EXITO\n')
      f.write('Fecha y hora de fin de experimentacion:     '
+str(fecha_fin))
      f.flush()
      f.close()
      g.close()
      h.close()
      k.close()



##    CUERPO DEL CODIGO.

def main():

      if(len(sys.argv)<3):
            print '\n\nPARAMETROS INCORRECTOS. \n' \
            'Los parametros deben ser:\n' \
            'Succion motor (50-100) \n' \
            'Duracion de la experimentacion (en muestras)\n'\
            'Tiempo de apertura de cada válvula (en segundos)'
            return 0

      print 'Algoritmo: MARTINELLI'
      print 'Ruta Fichero: ' +ruta_fichero
      print 'Ruta Fichero de datos 1: '+ruta_fichero_data1
      print 'Ruta Fichero de datos 2: '+ruta_fichero_data2

      #SUCCION MOTOR:   1-100%

      succion = float(sys.argv[1])
      if succion > 100 or succion < 50:
            print '\n\nPARAMETRO MOTOR INCORRECTO.\n' \
            'Los parametros deben ser:\n' \
            'Succion motor (50-100) \n' \
            'Duracion de la experimentacion (en muestras)\n'\
            'Tiempo de apertura de cada válvula (en segundos)'
            return 0
      else :
```

```
            print 'Succion motor al ' +str(succion)+ '%    Motor Pin PWM
: ' +motorPin

      #DURACION DEL EXPERIMENTO

      SAMPLES = float(sys.argv[2])
      if SAMPLES < 1:
            print '\n\nPARAMETRO TIEMPO EXPERIMENTACION.\n' \
            'Los parametros deben ser:\n' \
            'Succion motor (50-100) \n' \
            'Duracion de la experimentacion (en muestras)\n'\
            'Tiempo de apertura de cada válvula (en segundos)'
            return 0
      else :
            print 'Duracion de la experimentacion ' +str(SAMPLES)+ '
muestras'

      #TIEMPO DE APERTURA DE CADA VÁLVULA

      conmutacion = float(sys.argv[3])
      if conmutacion < SAMPLES:
            print '\n\nPARAMETRO TIEMPO EXPERIMENTACION.\n' \
            'Los parametros deben ser:\n' \
            'Succion motor (50-100) \n' \
            'Duracion de la experimentacion (en muestras)\n'
            return 0
      else :
            print 'Tiempo de apertura de cada electrovalvula '
+str(conmutacion)+ ' segundos\n'

      raw_input("Si es correcto presiona enter, sino ctr-c") #Se
comprueba por pantalla que los parametros son correctos

      #Se calcula de forma aleatoria la conmutacion de electrovalvulas
      vec_open_valve = numpy.random.randint(1,4,SAMPLES)
      print 'Conmutacion entre electrovalvulas:  ' +str(vec_open_valve)+
'\n'

      #Llamada al thread de medida de temperatura y humedad
      thread.start_new_thread(measure_tyh, (True,))

      #Se realiza el Setup de los puertos ADC, PWM, GPIO
      motor_start(succion)
      ADC.setup()
      PWM.start(heatPin2600,heat2600,20000,0)

      #Se crea el fichero de informacion de la experimentacion
      file_TGS2600(vec_open_valve,succion,heat2600,SAMPLES)

      print '\nComienza la adquisicion. Muestras inciales: '
+str(SAMPLESINICIO)+ '\n\n'
      f = open(ruta_fichero, "a")
      f.write('\n\nComienza la adquisicion. Muestras inciales: '
+str(SAMPLESINICIO)+ '\n\n')
      f.flush()

      #Calentamiento del sensor, se toman SAMPLESINICIO medidas antes de
comenzar la experimentacion
      i=1
      while i <= SAMPLESINICIO:
```

```
            time_martinelli_ini = time.time()
            samplesinicio_martinelli_TGS2600()
            time_martinelli_fin = time.time()
            instante_captura=datetime.now()
            time_martinelli = time_martinelli_fin-time_martinelli_ini
      #Duracion de los k=16 pulsos
            duracion_k.append(time_martinelli)
            print '\nSamplesinicio_Iteracion['+str(i)+']   Duracion de
los k=16 pulsos: '+str(time_martinelli)+' >> '+str(instante_captura)+'\n'
            wline_f = 'Samplesinicio_Iteracion['+str(i)+'] Duracion de
los k=16 pulsos: '+str(time_martinelli)+' >> '+str(instante_captura)+'\n'
            wline_k = str(i)+'      '+str(time_martinelli)+'
      '+str(instante_captura)+'\n'
            f = open(ruta_fichero, "a")
            k = open(ruta_fichero_data2, "a")
            f.writelines(wline_f)
            k.writelines(wline_k)
            f.flush()
            k.flush()
            i+=1

      #Comienza la experimentacion
      print '\n\nSE VAN A CAPTURAR: ' +str(SAMPLES)+' muestras\n\n'
      f.write('\n\nSE VAN A CAPTURAR: ' +str(SAMPLES)+' muestras\n\n')
      f.flush()

      iteracion = 0
      while iteracion <= SAMPLES-1:
            electrovalvula = vec_open_valve[iteracion]
            if electrovalvula == 1:
                  print 'ELECTROVALVULA: '+str(electrovalvula)+'
      METANOL\n'
                  f.write ('ELECTROVALVULA: '+str(electrovalvula)+'
      METANOL\n')
                  f.flush()
                  apertura(electrovalvula)
            elif electrovalvula == 2:
                  print 'ELECTROVALVULA: '+str(electrovalvula)+'
      ETANOL\n'
                  f.write ('ELECTROVALVULA: '+str(electrovalvula)+'
      ETANOL\n')
                  f.flush()
                  apertura(electrovalvula)
            elif electrovalvula == 3:
                  print 'ELECTROVALVULA: '+str(electrovalvula)+'
      BUTANOL\n'
                  f.write ('ELECTROVALVULA: '+str(electrovalvula)+'
      BUTANOL\n')
                  f.flush()
                  apertura(electrovalvula)

            tiempo_valve = 0

            while tiempo_valve <= conmutacion:
                  time_martinelli_ini = time.time()
                  martinelli_TGS2600()
                  time_martinelli_fin = time.time()
                  instante_captura=datetime.now()
                  time_martinelli = time_martinelli_fin-
time_martinelli_ini     #Duracion de los k=16 pulsos
```

```
                  duracion_k.append(time_martinelli)


                  print '\nIteracion['+str(iteracion)+']   Duracion de
los k=16 pulsos: '+str(time_martinelli)+' >> '+str(instante_captura)+'\n'
                  wline_f = 'Iteracion['+str(iteracion)+'] Duracion de
los k=16 pulsos: '+str(time_martinelli)+' >> '+str(instante_captura)+'\n'
                  wline_k = str(iteracion)+'    '+str(time_martinelli)+'
      '+str(instante_captura)+'\n'
                  f = open(ruta_fichero, "a")
                  k = open(ruta_fichero_data2, "a")
                  f.writelines(wline_f)
                  k.writelines(wline_k)
                  f.flush()
                  k.flush()

                  tiempo_valve += time_martinelli
                  print 'Tiempo acumulado = '+str(tiempo_valve)

            iteracion +=1

      #Terminar
      cierre()
      return 0



if __name__ == '__main__':
      try:
            main()
      except KeyboardInterrupt:
            print'Interrupted'
            cierre()
            sys.exit(0)
```

H.3. Código modulación regresión de temperatura

```
# -*- encoding: utf-8 -*- # Especificamos que nuestro archivo .py esta
codificado en UTF-8
#!/usr/bin/python

"""
      Plataforma de experimentacion

    SERGIO DE LA CRUZ GUTIERREZ

    ALGORITMO REGRESION TEMPERATURA
    Parametros:
            Succion motor (50-100)
            Temperatura Promedio TGS2600 (1-100)
            Tiempo de experimentacion (en minutos)
            Tiempo conmutacion electrovalvula (en segundos)

      Sentencia de ejecucion: tgs2600regresion.py succion heat tiempo
switch

"""

##    Declaracion de librerias a utilizar.

import Adafruit_BBIO.ADC as ADC
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.PWM as PWM
import Adafruit_DHT
import sys
import os
import time
import math
import random
import numpy
from scipy import stats
from datetime import datetime, date
import thread



##    ASIGNACION DE PUERTOS.

electrovalve1 = 'P8_10'       #Electrovalvula 1 (METANOL)
electrovalve2 = 'P8_12'       #Electrovalvula 2 (ETANOL)
electrovalve3 = 'P8_14'       #Electrovalvula 3 (BUTANOL)
electrovalve4 = 'P8_16'       #Electrovalvula 4 (AIRE)

motorPin = 'P9_21'            #Motor de succion

sensorPin2600 = 'P9_40'       #Sensor TGS2600 (Puerto AIN_1)
heatPin2600 = 'P9_22'         #Calentamiento sensor TGS2600 (Puerto
GPIO_2)

sensorTemp22 =  Adafruit_DHT.DHT22
```

```
Temp22 = 'P8_11'        #Pin de lectura de temperatura/Humedad con DHT22
(Puerto GPIO_45)


##    DECLARACION DE SALIDAS DEL SISTEMA.

GPIO.setup("P8_10",GPIO.OUT)
GPIO.setup("P8_12",GPIO.OUT)
GPIO.setup("P8_14",GPIO.OUT)
GPIO.setup("P8_16",GPIO.OUT)


##    DECLARACION DE VARIABLES DEL SISTEMA.

# Regresion.

NM=10       #Numero lecturas ADC
T=0.1             #Las NM lecturas se hacen en T segundos
tsub=T/NM;        #Subdivisiones de tiempo para las lecturas ADC
SLEEP=1           #Periodo de estabilizacion cuando se pone una nueva
temperatura
SAMPLESINICIO=10        #Capturas iniciales
TENDENCIA=5       #Tendencia de la modulacion de temperatura

x=[]        #Almacena el numero de muestra
concentTGS2600=[]       #Almacena las muestras de odorante
tempTGS2600=[]          #Almacena las temperaturas de calentamiento del
sensor

# Sensor TGS2600.

Rl_2600=27000 #Omh
Vc=5 #V

# Measure temperatura y humedad
SLEEP_tyh = 59


##    NOMBRE Y RUTA DE LOS FICHEROS DE SALIDA.

nameFile = "Regresion_TGS2600.txt"  #Fichero de informacion con
parámetros y lecturas de la experimentacion
nameFile_data = "Regresion_TGS2600.dat"   #Fichero de salida de datos
nameFile_tyh = "TyH_TGS2600.data"   #Fichero de informacion de
temperatura y humedad dela experimentacion
fileString = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime())+'_'
+nameFile
fileString_data =time.strftime("%a%d%b%Y-%HH%MM%SS",
time.localtime())+'_' +nameFile_data
fileString_tyh = time.strftime("%a%d%b%Y-%HH%MM%SS",
time.localtime())+'_' +nameFile_tyh
ahora=datetime.now()
ruta =
"/root/AMPLITUD/REGRESION/"+str(ahora.month)+"/"+str(ahora.day)+"/"
      #Ruta de salida
if not os.path.exists(ruta): os.makedirs(ruta) #Si la ruta no existe, se
crea
ruta_fichero = ruta.strip() + str(fileString)
ruta_fichero_data = ruta.strip() + str(fileString_data)
ruta_fichero_tyh = ruta.strip() + str(fileString_tyh)
```

```
## Creacion de los ficheros de datos y Temperatura/Humedad del
experimento.

g=open(ruta_fichero_data, "w+")
h = open(ruta_fichero_tyh, "w+")


##    FUNCION de creacion del fichero de información del experimento.
Sensor TGS2600 (modo escritura) y cabecera

def file_TGS2600(vec_open_valve,succion,heat2600,tiempo):

      f = open(ruta_fichero, "w+")
      f.write
('////////////////////////////////////////////////////////////////////\n
')
      f.write ('\n\nPlataforma de experimentacion: \n\nSERGIO DE LA CRUZ
GUTIERREZ\n\n')
      f.write ('Sensor TGS2600\n')
      f.write ('Algoritmo REGRESION TEMPERATURA\n')
      date=time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime())
      f.write ('Fecha y hora de inicio: ' + str(date) + '\n')
      f.write ('Ruta del fichero: ' + str(ruta) + '\n')
      f.write ('Nombre del fichero: ' + str(nameFile) + '\n')
      f.write ('Nombre del fichero de datos: ' + str(nameFile_data) +
'\n')
      f.write ('Nombre del fichero de TyH: ' + str(nameFile_tyh) + '\n')
      f.write ('Electrovalvula (1-METANOL, 2-ETANOL, 3-BUTANOL, 4-AIRE )
\n')
      f.write ('Conmutacion entre electrovalvulas: '
+str(vec_open_valve)+ '\n')
      f.write ('\n')
      f.write ('Parametros para la experimetacion, se introducen como
argumentos \n')
      f.write ('Succion motor (50-100)    >>> '+ str(succion)+ '%\n')
      f.write ('Temperatura Promedio TGS2600 (1-100) >>>
'+str(heat2600)+'%      Pin PWM : ' +str(heatPin2600)+ '\n\n')
      f.write ('Duracion del esperimento: ' +str(tiempo)+ ' minutos\n')
      f.write
('////////////////////////////////////////////////////////////////////\n
\n')
      f.write ('  CAPTURA DE DATOS:\n\n')



## DECLARACION DE FUNCIONES DEL SISTEMA.

#     Apertura electrovalvula

def apertura(electrovalvula):
      if electrovalvula == 1:
            GPIO.output(electrovalve1, GPIO.HIGH)
            GPIO.output(electrovalve2, GPIO.LOW)
            GPIO.output(electrovalve3, GPIO.LOW)
            GPIO.output(electrovalve4, GPIO.LOW)
      elif electrovalvula == 2:
            GPIO.output(electrovalve1, GPIO.LOW)
            GPIO.output(electrovalve2, GPIO.HIGH)
```

```
            GPIO.output(electrovalve3, GPIO.LOW)
            GPIO.output(electrovalve4, GPIO.LOW)
      elif electrovalvula == 3:
            GPIO.output(electrovalve1, GPIO.LOW)
            GPIO.output(electrovalve2, GPIO.LOW)
            GPIO.output(electrovalve3, GPIO.HIGH)
            GPIO.output(electrovalve4, GPIO.LOW)
      elif electrovalvula == 4:
            GPIO.output(electrovalve1, GPIO.LOW)
            GPIO.output(electrovalve2, GPIO.LOW)
            GPIO.output(electrovalve3, GPIO.LOW)
            GPIO.output(electrovalve4, GPIO.HIGH)


#     Arranque motor

def motor_start(succion):
      PWM.start(motorPin,succion)


#     Funcion de medida de temperatura y humedad

def measure_tyh(tiempo):

      j=0
      tiempo=round(tiempo)
      time.sleep(9)
      #Lectura humedad y temperatura
      while j<tiempo:
            tick_HT = time.time()
            humidity, temperature = Adafruit_DHT.read_retry(sensorTemp22,
Temp22)
            instante = datetime.now()
            tack_HT=time.time()
            t_HT=tack_HT-tick_HT


            #Cuanto duerme en funcion de lo que tarde en H y T
            if t_HT > SLEEP_tyh:
                  print "Tiempo medicion H y T > SLEEP:", t_HT
            else:
                  print "Tiempo medicion H y T:", t_HT


                  if humidity is not None  and temperature is not None:
                        print'\nSensor DHT22: ' +str(sensorTemp22)
                        print'>>> '+str(instante)+'   Temp = '
+str(temperature)+ '    Humidity = ' +str(humidity)+'\n'
                        h = open(ruta_fichero_tyh, "a")
                        wline_h=str(instante)+' '+str(temperature)+'
      '+str(humidity)+'\n'
                        h.writelines(wline_h)
                        h.flush()

                  else :
                        print 'Failed to get reading, Try again!'
                        h = open(ruta_fichero_tyh, "a")
                        wline_h=str(instante)+' '+str(temperature)+'
      '+str(humidity)+'\n'
                        h.writelines(wline_h)
```

```
                        h.flush()
                tack=time.time()
                time.sleep(SLEEP_tyh-(tack-tick_HT))
                j+=1


#     Samplesinicio Regresion TGS2600

def ini_regresion_TGS2600(count, heat2600):

      random.seed()
      temperature_TGS2600 = heat2600;
      PWM.set_duty_cycle(heatPin2600, temperature_TGS2600)

      #Lectura nariz

      value=0
      queda=0
      residuo=0
      ADC.read_raw(sensorPin2600) #la primera medida es erronea por el
bug
      for i in range(NM):
            value += ADC.read_raw(sensorPin2600)
            r=random.uniform(0, tsub)
            time.sleep(r)
            residuo += (tsub-r)
      time.sleep(residuo);
      valueTGS2600=value/NM
      instante_captura=datetime.now()

      x.append(count)
      concentTGS2600.append(valueTGS2600)
      tempTGS2600.append(temperature_TGS2600)

      #Se calcula el valor de la resistencia interna del sensor
      RsTGS2600=((Vc*Rl_2600)/(valueTGS2600/1000.))-Rl_2600

      #Escritura por pantalla de la lectura
      print 'Muestra Regresion_ini['+str(count)+']\n>> Valor:
'+str(valueTGS2600)+'mV >> Rs: '+str(RsTGS2600)+'     >> Temperatura:
'+str(temperature_TGS2600)+'  >> '+str(instante_captura)+'\n'

      #Se escriben los datos en el fichero
      f = open(ruta_fichero, "a")
      g = open(ruta_fichero_data, "a")
      wline_g=str(count)+'    '+str(valueTGS2600)+'   '+str(RsTGS2600)+'
      '+str(temperature_TGS2600)+'  '+str(instante_captura)+'\n'
      wline_f='Muestra_ini['+str(count)+']
      Valor:'+str(valueTGS2600)+'mV -- Rs: '+str(RsTGS2600)+'     --
Temperatura: '+str(temperature_TGS2600)+' >>'+str(instante_captura)+'\n'
      g.writelines(wline_g)
      g.flush()
      f.writelines(wline_f)
      f.flush()


#     Regresion TGS2600

def regresion_TGS2600(count, heat2600):
```

```
	value=0
	queda=0
	residuo=0
	ADC.read_raw(sensorPin2600) #la primera medida es erronea por el
bug
	for i in range(NM):
		value += ADC.read_raw(sensorPin2600)
		r=random.uniform(0, tsub)
		time.sleep(r)
		residuo += (tsub-r)
	time.sleep(residuo); #sleep+(tsub*NM)+T+residuo=1seg
	valueTGS2600=value/NM
	instante_captura=datetime.now()

	#Adaptacion temperatura
	slope, intercept, r_value, p_value, std_err1 =
stats.linregress(x[(count-SAMPLESINICIO):(count-
1)],concentTGS2600[(count-SAMPLESINICIO):(count-1)])
	temperature_TGS2600 = heat2600 - (slope*TENDENCIA)

	if temperature_TGS2600 < 10.0:
		temperature_TGS2600 = 10.0
	elif temperature_TGS2600 >90.0:
		temperature_TGS2600 = 90.0

	#Reset de setup PWM
	PWM.set_duty_cycle(heatPin2600, temperature_TGS2600)

	x.append(count)
	concentTGS2600.append(valueTGS2600)
	tempTGS2600.append(temperature_TGS2600)

	#Se calcula el valor de la resistencia interna del sensor
	RsTGS2600=((Vc*Rl_2600)/(valueTGS2600/1000.))-Rl_2600

	#Escritura por pantalla de la lectura
	print 'Muestra Regresion['+str(count)+']\n>> Valor:
'+str(valueTGS2600)+'mV >> Rs: '+str(RsTGS2600)+'    >> Temperatura:
'+str(temperature_TGS2600)+' >> '+str(instante_captura)+'\n'

	#Se escriben los datos en el fichero
	f = open(ruta_fichero, "a")
	g = open(ruta_fichero_data, "a")
	wline_g=str(count)+'    '+str(valueTGS2600)+'   '+str(RsTGS2600)+'
	'+str(temperature_TGS2600)+' '+str(instante_captura)+'\n'
	wline_f='Muestra['+str(count)+']    Valor:'+str(valueTGS2600)+'mV --
Rs: '+str(RsTGS2600)+' --Temperatura: '+str(temperature_TGS2600)+'
	>>'+str(instante_captura)+'\n'
	g.writelines(wline_g)
	g.flush()
	f.writelines(wline_f)
	f.flush()


#	Cierre

def cierre():

	PWM.stop(heatPin2600)
```

```
      PWM.stop(motorPin)
      PWM.cleanup()
      thread.exit()
      fecha_fin=datetime.now()
      print('\nEXPERIMENTO FINALIZADO CON EXITO')
      print 'Experimento terminado: '+str(fecha_fin)
      f = open(ruta_fichero, "a")
      f.write('\nEXPERIMENTO FINALIZADO CON EXITO\n')
      f.write('Fecha y hora de fin de experimentacion:     '
+str(fecha_fin))
      f.flush()
      f.close()
      g.close()
      h.close()


##    CUERPO DEL CODIGO.

def main():

      if(len(sys.argv)<5):
            print '\n\nPARAMETROS INCORRECTOS. \n' \
            'Los parametros deben ser:\n' \
            'Succion motor (50-100) \n' \
            'Temperatura Promedio TGS2600 (1-100) \n' \
            'Tiempo de experimentacion (en minutos) \n'\
            'Tiempo conmutacion electrovalvula (en segundos) \n'
            return 0

      print '\nSensor: TGS2600    Pin ADC: ' +sensorPin2600
      print 'Algoritmo: REGRESION TEMPERATURA'
      print 'Ruta Fichero: ' +ruta_fichero
      print 'Ruta Fichero de datos: '+ruta_fichero_data

      #SUCCION MOTOR:   50-100%

      succion = float(sys.argv[1])
      if succion > 100 or succion < 50:
            print '\n\nPARAMETRO MOTOR INCORRECTO.\n' \
            'Los parametros deben ser:\n' \
            'Succion motor (50-100) \n' \
            'Temperatura Promedio TGS2600 (1-100) \n' \
            'Tiempo de experimentacion (en minutos) \n'\
            'Tiempo conmutacion electrovalvula (en segundos) \n'
            return 0
      else :
            print 'Succion motor al ' +str(succion)+ '%    motor Pin PWM
: ' +motorPin

      #TEMPERATURA PROMEDIO TGS2600:     1-100%

      heat2600 = float(sys.argv[2])
      if heat2600 > 100:
            print '\n\nPARAMETRO CALENTAMIENTO PROMEDIO INCORRECTO.\n' \
            'Los parametros deben ser:\n' \
            'Succion motor (50-100) \n' \
            'Temperatura Promedio TGS2600 (1-100) \n' \
            'Tiempo de experimentacion (en minutos) \n'\
            'Tiempo conmutacion electrovalvula (en segundos) \n'
            return 0
```

```
      else :
            print 'Calentamiento promedio: ' +str(heat2600)+ '%
TGS2600 Pin PWM : ' +heatPin2600

      #DURACION DEL EXPERIMENTO

      tiempo = float(sys.argv[3])
      if tiempo < 1:
            print '\n\nPARAMETRO TIEMPO EXPERIMENTACION.\n' \
            'Los parametros deben ser:\n' \
            'Succion motor (50-100) \n' \
            'Temperatura Promedio TGS2600 (1-100) \n' \
            'Tiempo de experimentacion (en minutos) \n'\
            'Tiempo conmutacion electrovalvula (en segundos) \n'
            return 0
      else :
            print 'Tiempo de experimentación: ' +str(tiempo)+ '
minutos\n'

      #TIEMPO CONMUTACION ELECTROVALVULA

      switch = float(sys.argv[4])
      if switch <1:
            print '\n\nPARAMETRO TIEMPO CONMUTACION ELECTROVALVULA
INCORRECTO.\n' \
            'Los parametros deben ser:\n' \
            'Succion motor (50-100) \n' \
            'Temperatura Promedio TGS2600 (1-100) \n' \
            'Tiempo de experimentacion (en minutos) \n'\
            'Tiempo conmutacion electrovalvula (en minutos) \n'
            return 0
      else :
            print 'Tiempo conmutacion electrovalvula: ' +str(switch)+'
      segundos'


      raw_input("Si es correcto presiona enter, sino ctr-c\n") #Se
comprueba por pantalla que los parametros son correctos

      #Se calcula de forma aleatoria la conmutacion de electrovalvulas
      vec_open_valve = numpy.random.randint(1,4,tiempo*60/switch)
      print 'Conmutacion entre electrovalvulas:  ' +str(vec_open_valve)+
'\n'

      #Llamada al thread de medida de temperatura y humedad
      thread.start_new_thread(measure_tyh, (tiempo,))

      #Se realiza el Setup de los puertos ADC, PWM, GPIO
      motor_start(succion)
      ADC.setup()
      PWM.start(heatPin2600,heat2600,20000,0)

      #Se crea el fichero de informacion de la experimentacion
      file_TGS2600(vec_open_valve,succion,heat2600,tiempo)

      print '\n\nComienza la adquisicion. Muestras inciales: '
+str(SAMPLESINICIO)+ '\n\n'
      f = open(ruta_fichero, "a")
      f.write('\n\nComienza la adquisicion. Muestras inciales: '
+str(SAMPLESINICIO)+ '\n\n')
```

```
	f.flush()

	#Calentamiento del sensor, se toman SAMPLESINICIO medidas antes de
comenzar la experimentacion
	count=1
	while count<=SAMPLESINICIO:
		tick_reg=time.time()
		ini_regresion_TGS2600(count, heat2600)
		count+=1
		tack_reg=time.time()
		time.sleep(SLEEP-(tack_reg-tick_reg))

	print '\n\nSe van a capturar: ' +str(tiempo*60)+' muestras\n\n'
	f.write('\n\nSe van a capturar: ' +str(tiempo*60)+' muestras\n\n')
	f.flush()


	#Comienza la experimentacion

	i=0

	while count<=(tiempo*60) + SAMPLESINICIO :

		electrovalvula = vec_open_valve[i]

		if electrovalvula == 1:
			print 'ELECTROVALVULA: '+str(electrovalvula)+' METANOL
\n'
		elif electrovalvula == 2:
			print 'ELECTROVALVULA: '+str(electrovalvula)+' ETANOL
\n'
		elif electrovalvula == 3:
			print 'ELECTROVALVULA: '+str(electrovalvula)+' BUTANOL
\n'

		apertura(electrovalvula)

		j=0
		while j<switch:
			tick_regresion=time.time()
			regresion_TGS2600(count, heat2600)
			count+=1
			j+=1
			tack_regresion=time.time()
			time.sleep(SLEEP-(tack_regresion-tick_regresion))

		i+=1

	#Terminar
	cierre()
	return 0

if __name__ == '__main__':
	try:
		main()
	except KeyboardInterrupt:
		print'Interrupted'
		cierre()
		sys.exit(0)
```

H.4. Código Limpieza circuito electroválvulas

```
# -*- encoding: utf-8 -*- # Especificamos que nuestro archivo .py esta
codificado en UTF-8
#!/usr/bin/python

"""
      Plataforma de experimentacion

    SERGIO DE LA CRUZ GUTIERREZ

    ALGORITMO LIMPIEZA
    Parametros:
            Electrovalvula (0-TODAS, 1-METANOL, 2-ETANOL, 3-BUTANOL)
            Tiempo de la limpieza (en minutos)


      Sentencia de ejecucion: limpiezs.py electrovalvula tiempo

"""


##    Declaracion de librerias a utilizar.

import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.PWM as PWM
import time


##    ASIGNACION DE PUERTOS.

electrovalve1 = 'P8_10'       #Electrovalvula 1 (METANOL)
electrovalve2 = 'P8_12'       #Electrovalvula 2 (ETANOL)
electrovalve3 = 'P8_14'       #Electrovalvula 3 (BUTANOL)
electrovalve4 = 'P8_16'       #Electrovalvula 4 (AIRE)

motorPin = 'P9_21'            #Motor de succion


##    DECLARACION DE SALIDAS DEL SISTEMA.

GPIO.setup("P8_10",GPIO.OUT)
GPIO.setup("P8_12",GPIO.OUT)
GPIO.setup("P8_14",GPIO.OUT)
GPIO.setup("P8_16",GPIO.OUT)


##    NOMBRE Y RUTA DE LOS FICHEROS DE SALIDA.

nameFile="Limpieza_TGS2600.txt"     #Fichero de informacion con
parámetros y lecturas de la experimentacion
fileString= time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime())+'_'
+nameFile
ruta = "/root/LIMPIEZA"+str(ahora.month)+"/"+str(ahora.day)+"/"
      #Ruta de salida
if not os.path.exists(ruta): os.makedirs(ruta) #Si la ruta no existe, se
crea
```

```
ruta_fichero = ruta.strip() + str(fileString)

##    FUNCION de creacion del fichero de información del experimento.
Sensor TGS2600 (modo escritura) y cabecera

def file_limpieza(electrovalvula,tiempo):

      f = open(ruta_fichero, "w+")
      f.write ('\n\nPlataforma de experimentacion: \n\nSERGIO DE LA CRUZ
GUTIERREZ\n\n')
      f.write ('Sensor TGS2600\n')
      f.write ('Algoritmo Limpieza\n')
      date=time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime())
      f.write ('Fecha y hora de inicio: ' + str(date) + '\n')
      f.write ('Ruta del fichero: ' + str(ruta) + '\n')
      f.write ('Nombre del fichero: ' + str(nameFile) + '\n')
      f.write ('Electrovalvula (1-METANOL, 2-ETANOL, 3-BUTANOL, 0-TODAS )
\n')
      f.write ('\n')
      f.write ('Parametros para la experimetacion, se introducen como
argumentos \n')
      f.write ('Electrovalvula/s a limpiar:    '+str(electrovalvula))
      f.write ('Duracion del esperimento: ' +str(tiempo)+ ' minutos\n')
      f.write
('////////////////////////////////////////////////////////////////////\n
\n')


## DECLARACION DE FUNCIONES DEL SISTEMA.

# Apertura electrovalvula

def apertura(electrovalvula):
      if electrovalvula == 1:
            GPIO.output(electrovalve1, GPIO.HIGH)
            GPIO.output(electrovalve2, GPIO.LOW)
            GPIO.output(electrovalve3, GPIO.LOW)
            GPIO.output(electrovalve4, GPIO.LOW)
      elif electrovalvula == 2:
            GPIO.output(electrovalve1, GPIO.LOW)
            GPIO.output(electrovalve2, GPIO.HIGH)
            GPIO.output(electrovalve3, GPIO.LOW)
            GPIO.output(electrovalve4, GPIO.LOW)
      elif electrovalvula == 3:
            GPIO.output(electrovalve1, GPIO.LOW)
            GPIO.output(electrovalve2, GPIO.LOW)
            GPIO.output(electrovalve3, GPIO.HIGH)
            GPIO.output(electrovalve4, GPIO.LOW)
      elif electrovalvula == 4:
            GPIO.output(electrovalve1, GPIO.HIGH)
            GPIO.output(electrovalve2, GPIO.HIGH)
            GPIO.output(electrovalve3, GPIO.HIGH)
            GPIO.output(electrovalve4, GPIO.HIGH)

# Arranque motor

def motor_start(succion):
      PWM.start(motorPin,succion)

# Parar motor
```

```
def motor_stop():
      PWM.stop(motorPin)

# Cierre

def cierre():

      PWM.stop(motorPin)
      PWM.cleanup()
      fecha_fin=date
      print('\n\nLIMPIEZA REALIZADA CON EXITO')
      print 'Limpieza terminada:    '+str(fecha_fin)
      f = open(ruta_fichero, "a")
      f.write('\n\nLIMPIEZA REALIZADA CON EXITO\n')
      f.write('Fecha y hora de fin de la limpieza:    ' +str(fecha_fin))
      f.flush()
      f.close()

##    CUERPO DEL CODIGO.

def main():

      if(len(sys.argv)<2):
            print '\n\nPARAMETROS INCORRECTOS. \n' \
            'Los parametros deben ser:\n' \
            'Electrovalvula (0-TODAS, 1-METANOL, 2-ETANOL, 3-BUTANOL) \n'
\
            'Tiempo de la limpieza (en minutos)\n'
            return 0

      print 'Ruta Fichero: ' +ruta_fichero
      print 'Ruta Fichero de datos: '+ruta_fichero_data
      print 'Algoritmo: Limpieza'

      #ELECTROVALVULA:  1-METANOL   2-ETANOL    3-BUTANOL   0-TODAS

      electrovalvula = float(sys.argv[1])
      if electrovalvula > 4:
            print '\n\nPARAMETRO CALENTAMIENTO PROMEDIO INCORRECTO.\n' \
            'Los parametros deben ser:\n' \
            'Electrovalvula (0-TODAS, 1-METANOL, 2-ETANOL, 3-BUTANOL) \n'
\
            'Tiempo de la limpieza (en minutos)\n'
            return 0
      elif electrovalvula == 0:
            print 'Electrovalvula: 0-TODAS'
      elif electrovalvula == 1:
            print 'Electrovalvula: 1-METANOL'
      elif electrovalvula == 2:
            print 'Electrovalvula: 2-ETANOL'
      elif electrovalvula == 3:
            print 'Electrovalvula: 3-BUTANOL'

      #DURACION DEL EXPERIMENTO

      tiempo = float(sys.argv[3])
      if tiempo < 1:
            print '\n\nPARAMETRO TIEMPO EXPERIMENTACION.\n' \
            'Los parametros deben ser:\n' \
```

```
            'Electrovalvula (0-TODAS, 1-METANOL, 2-ETANOL, 3-BUTANOL) \n'
\
            'Tiempo de la limpieza (en minutos)\n'
            return 0
      else :
            print 'Tiempo de experimentación: ' +str(tiempo)+ '\n'

      raw_input("Si es correcto presiona enter, sino ctr-c") #Se
comprueba por pantalla que los parametros son correctos

      file_limpieza()

      if electrovalvula == 0:
            time=tiempo/4
            i=0
            for i <= 4:
                  motor_start(100)
                  apertura(i)
                  time.sleep(time)
                  motor_stop()
                  i+=1

      elif electrovalvula == 1:
            motor_start(100)
            apertura(electrovalvula)
            time.sleep(tiempo)
            motor_stop()

      elif electrovalvula == 2:
            motor_start(100)
            apertura(electrovalvula)
            time.sleep(tiempo)
            motor_stop()

      elif electrovalvula == 3:
            motor_start(100)
            apertura(electrovalvula)
            time.sleep(tiempo)
            motor_stop()

      #Terminar
      cierre()
      return 0

if __name__ == '__main__':
      main()
```

Presupuesto

1) Ejecución Material

- Compra de ordenador personal (Software incluido) .... 800 €
- Material de oficina ..... 50 €
- Sistema embebido ..... 60 €
- Electroválvulas ..... 200 €
- Material de montaje ..... 40 €
- Motor de succión ..... 55 €
- Sensores ..... 25 €
- Transformadores ..... 35 €
- Resto de componentes ..... .50 €
- Total, de ejecución material ..... 1.315 €

2) Gastos generales

- 16 % sobre Ejecución Material ..... 210,4 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material ..... 78,9 €

4) Honorarios Proyecto

- 2000 horas a 15 € / hora ..... 30000 €

5) Material fungible

- Gastos de impresión ..... 80 €
- Encuadernación ..... 30 €

6) Subtotal del presupuesto

- Subtotal Presupuesto..... 31.714,3 €

7) I.V.A. aplicable

- 21% Subtotal Presupuesto ..... 6.660 €

8) Total, presupuesto

- Total, Presupuesto..... 38.374,3 €

Madrid, juLio de 2016
El Ingeniero Jefe de Proyecto

Fdo.: Sergio de la Cruz Gutiérrez
Ingeniero de Telecomunicación

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un Desarrollo de una plataforma para discriminación de odorantes mediante técnicas de modulación dinámica. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en

el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras, así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata" y anteriormente llamado" Presupuesto de Ejecución Material" que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones, así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.

Figura H-1: Logo EPS