Escuela Politécnica Superior de Jaén
Grado en Ingeniería Electrónica Industrial

UNIVERSIDAD DE JAÉN
ESCUELA POLITÉCNICA SUPERIOR DE JAÉN

Trabajo Fin de Grado

DESARROLLO SOFTWARE DE UNA NARIZ ELECTRÓNICA PARA LA DISCRIMINACIÓN DE ACEITES

Alumno: Arturo López Riquelme

Tutor: Prof. D. Diego Manuel Martínez Gila

Prof. D. Pablo Cano Marchal

Dpto.: Electrónica

Junio, 2018

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Electrónica

Don Diego Manuel Martínez Gila, tutor del Trabajo Fin de Grado titulado *Desarrollo software de una nariz electrónica para la discriminación de aceites*, que presenta Arturo López Riquelme, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, junio de 2018

El alumno: Los tutores:

Arturo López Riquelme Diego Manuel Martínez Gila

Pablo Cano Marchal

RESUMEN

La cata del aceite de oliva virgen está sujeta a cierta subjetividad y es difícil determinar la calidad real del aceite analizando solo sus características organolépticas. La nariz electrónica, un proyecto multidisplicinar que se compone de tres tomos, Mecánica, Electrónica Hardware y Software, utiliza sensores de gas para captar componentes volátiles de la muestra y así poder hallar la huella sensorial del aceite de manera objetiva. Este tomo trata el desarrollo de la Electrónica Software.

ABSTRACT

The tasting of virgin olive oil is subject to certain subjectivity and it is difficult to determine the actual quality of the oil by analyzing only its organoleptic characteristics. The electronic nose, a project that consists of three volumes, Mechanics, Electronics Hardware and Software, uses gas sensors to capture volatile components of the sample and thus be able to find the sensory footprint of the oil objectively. This volume deals with the development of Electronic Software.

Palabras clave: Nariz Electrónica, Industria 4.0, Aceite de Oliva Virgen, Virgen Extra, Calidad, Java, Raspberry Pi, Panel de Cata, Volátiles

A mis compañeros.

A mi familia.

A mis profesores.

Desarrollo software de una nariz electrónica para la discriminación de aceites

Índice

Lista de tablas

Lista de figuras

Lista de ilustraciones

Definiciones, Acrónimos y Abreviaturas

IoT	Internet of things
AOVE	Aceite de Oliva Virgen Extra
spin-off	Empresa nacida como extensión de otra
PCB	Printed Circuit Board
I2C	Inter-Integrated Circuit
JVM	Java Virtual Machine
JDK	Java Development Kit
JRE	Java Runtime Environment
IDE	Entorno de Desarrollo Integrado
SSH	Secure Shell
csv	comma separated values

Contenido del CD

Memoria formato digital.

Scripts de Java utilizados.

1. INTRODUCCIÓN

La iniciativa de desarrollar una nariz electrónica nace en el contexto de la investigación en el sector agro llevada a cabo durante muchos años por el grupo de investigación de Robótica, Automática y Visión por computador. El desarrollo software del proyecto es una de las líneas de trabajo que han sido resueltas para obtener como resultado una nariz electrónica completamente diseñada en este grupo de investigación de la Universidad de Jaén.

Las otras dos patas del proyecto han sido, el desarrollo hardware y el desarrollo mecánico. Ambas fundamentales para que el software tuviera cabida y utilidad. La consecución final de las tres líneas de trabajo confluye en la nariz electrónica.

El software está basado en la plataforma hardware Raspberry Pi y su enlace con el lenguaje de programación Java. Se quería dotar a la nariz electrónica de una interfaz de comunicación e interacción con el operario/usuario, que fuera capaz además de realizar la obtención y procesamiento de datos, la presentación de resultados de distinta formas y el sistema de control interno para el correcto funcionamiento de la unidad.

Una vez que ha sido obtenido un producto industrial, la nariz tiene muchos campos de aplicación: Industria alimentaria, detector de gases, detección de sustancias ilegales, etc. Sin embargo, en este caso se ha diseñado la nariz electrónica para la discriminación de la calidad del aceite en función del análisis de sus componentes volátiles

¿Por qué hacer un trabajo de fin de grado en la Universidad de Jaén dirigido al sector oleícola? Los datos del curso 2017 sobre el olivar en España, que recoge el Ministerio de Agricultura y Pesca, Alimentación y Medio Ambiente despejan las dudas.

Cultivo	Rendimiento			Producción total	Destino de la producción (toneladas)	
	De la superficie en producción (kg/ha)		De árboles diseminados (kg/árbol)		Aceituna para aderezo	Aceituna para almazara
	Secano	Regadío		(toneladas)		
Olivar de aceituna de mesa	2,330	5,307	4	485,197	332,586	152,611
Olivar de aceituna de almazara	1,868	4,959	6	5,825,016	212,638	5,612,378
Olivar total	**1,900**	**4,981**	**5**	**6,310,213**	**545,224**	**5,764,989**

Tabla 1.1 Resumen nacional de rendimiento y producción

Producción según clases	Cantidad (toneladas)
Extra	637,679
Virgen	112,477
Lampante	70,186
Total	**1,183,136**

Tabla 1.2 Datos de producción

Provincias y Comunidades Autónomas	Superficie en plantación regular (hectáreas)					Arboles diseminados (número)	Rendimiento			Producción (toneladas)
	Total			En producción			Superficie en producción (kg/ha)		Arboles diseminados (kg/árbol)	
	Secano	Regadío	Total	Secano	Regadío		Secano	Regadío		
Almería	7,056	14,162	21,218	5,969	12,975	–	413	5,891	–	78,896
Cádiz	21,392	2,277	23,669	20,607	1,794	–	1,900	4,338	–	46,936
Córdoba	293,305	54,228	347,533	286,528	47,697	–	3,498	6,284	–	1,302,034
Granada	126,351	70,046	196,397	119,584	67,571	2,560	1,902	3,802	28	484,406
Huelva	24,635	4,651	29,286	24,204	4,256	–	848	2,150	–	29,675
Jaén	332,539	249,888	582,427	330,710	248,703	–	1,230	5,000	–	1,650,288
Málaga	113,203	12,429	125,632	109,246	11,804	–	2,560	5,150	–	340,461
Sevilla	131,172	42,956	174,128	119,156	32,124	–	2,725	7,450	–	564,024
ANDALUCÍA	**1,049,653**	**450,637**	**1,500,290**	**1,016,004**	**426,924**	**2,560**	**2,267**	**5,138**	**28**	**4,496,720**
ESPAÑA	**1,841,047**	**546,785**	**2,387,832**	**1,754,170**	**513,627**	**54,393**	**1,868**	**4,959**	**6**	**5,825,016**

Tabla 1.3 Análisis provincial de superficie, rendimiento y producción. Aceituna de almazara

Provincias y Comunidades Autónomas	Extra	Virgen	Lampante	Total
Almería	14,480	1,448	161	16,089
Cádiz	5,655	1,740	1,305	8,700
Córdoba	207,146	18,277	18,277	243,700
Granada	103,883	3,079	1,453	108,415
Huelva	6,016	267	402	6,685
Jaén	–	–	–	360,000
Málaga	40,390	10,386	6,924	57,700
Sevilla	62,220	31,110	10,370	103,700
ANDALUCÍA	**439,790**	**66,307**	**38,892**	**904,989**
ESPAÑA	**637,679**	**112,477**	**70,186**	**1,183,136**

Tabla 1.4 Análisis provincial de la producción según clases (toneladas)

El sector del aceite tiene un impacto enorme en nuestro país, en nuestra comunidad autónoma y más si cabe en nuestra provincia, Jaén. La investigación en este sector se cimienta en tres pilares básicos:

- La escasa automatización en la industria de almazaras y la necesidad de cambiar esta situación.
- La gran producción de aceite y las ventajas que supondría avanzar tecnológicamente en su proceso de obtención.
- El fraude que, debido al factor humano, se da en la olivicultura.

1.1. Smart Sensors

Este término se ha adoptado recientemente para dispositivos electrónicos que utilizan información de su alrededor, de su entorno, para ser capaces de crear ambientes inteligentes. Se posicionan como los proveedores de información para la industria 4.0, la cual lleva implícito un cambio radical y nuevos desafíos originados por la mayor flexibilidad e individualización de los procesos de fabricación.

Otro término muy ligado es *'The Internet of Things'* (IoT). Se trata de hacer la vida más simple y más excitante para los consumidores, interconectando el mundo alrededor de ellos. En el mundo del IoT, los Smart Sensors hacen de intermediarios entre el usuario y la multitud de dispositivos que manejamos. En todas las aplicaciones que se desarrollan para IoT, los sensores constituyen el elemento indispensable para trasladar el mundo físico al mundo digital.

Figura 1.1 Eficiencia industrial en la era IoT

Con los sensores Smart se abre una puerta para la optimización de la eficiencia en el procesamiento y la recogida de datos en infraestructuras IoT, todo ello con la llegada de una simbiosis entre sensor y microcontrolador. Los Smart Sensors añaden un nivel de sofisticación superior en la eficiencia de los procesos de comunicación, tienen la capacidad de transformar los sectores industriales, incluso los ámbitos domésticos. Un sistema inteligente de sensor + microcontrolador es capaz computacionalmente de procesar, interpretar y tomar decisiones de manera aislada, basándose en parámetros físicos medibles.

Una vez que un sistema es capaz de tomar sus propias decisiones, se llega al concepto de inteligencia artificial (IA), machine learning, deep learning, redes neuronales, etc. Una nueva revolución industrial tendrá todos los protagonistas citados anteriormente, que darán lugar a plataformas más avanzadas de las que conocemos actualmente. (Gemelli, 2017)

Y antes de cerrar con el apartado, un apunte sobre la ciberseguridad. Será de vital importancia garantizar la seguridad y estabilidad de los sistemas conectados, ya que dada la proliferación de éstos, resultarán víctimas de ciberataques y serán elementos informáticos vulnerables, como lo son ahora nuestros ordenadores o servicios web.

1.2. Motivación

Las razones que llevan a un grupo de investigación de larga trayectoria a apostar por un proyecto de este tipo no son meras ilusiones de crear una máquina capaz de oler. Existe un trasfondo enmarañado de justificaciones y soluciones científicas a un problema de interés. La nariz electrónica no viene más que a tratar de poner tierra de por medio entre la sensorialidad humana y su juicio de valor del aceite, y el resultado electrónico que esta emita. Es el conflicto entre la subjetividad del panel de catas y la objetividad de la nariz electrónica en el objetivo común de la determinación de la calidad del aceite.

Pero para ir desglosando poco a poco la motivación de este proyecto, explicaremos los puntos principales en los que se basa su nacimiento.

1.2.1. Proceso de elaboración AOVE

El proceso de obtención del aceite siempre ha sido similar. Recientemente se ha requerido incluir modificaciones, nuevos métodos de mecanización y automatización que agilizan el proceso. La elaboración del AOVE es un proceso mecánico, se basa en exprimir las aceitunas para obtener su zumo. En la provincia de Jaén la actividad

industrial principal es este proceso de elaboración, el cual aporta a la economía una gran cantidad de beneficios. (Ver esquema en Figura 1.2)

En primer lugar se recoge el fruto, la aceituna, directamente del árbol. El olivo es un árbol que se encuentra a lo largo y ancho de toda la región. La recogida de aceituna es el primer paso. La campaña se inicia cuando la aceituna está en su punto óptimo de maduración. El momento óptimo de maduración se considera cuando la mayoría de la aceituna está cambiando de color. Normalmente la época de recolección de la aceituna se lleva a cabo desde finales de octubre hasta finales de enero o principios de febrero. Es en los meses de noviembre y diciembre cuando la aceituna recolectada es de alta calidad y da origen a aceites virgen extra. Para que el aceite se considere bueno, la materia prima de la recolección también debe serlo. Esto es, hay que evitar recolectar los frutos dañados o que hayan tocado el suelo. En cuestiones de calidad se distingue en muchas ocasiones entre aceituna de suelo y vuelo.

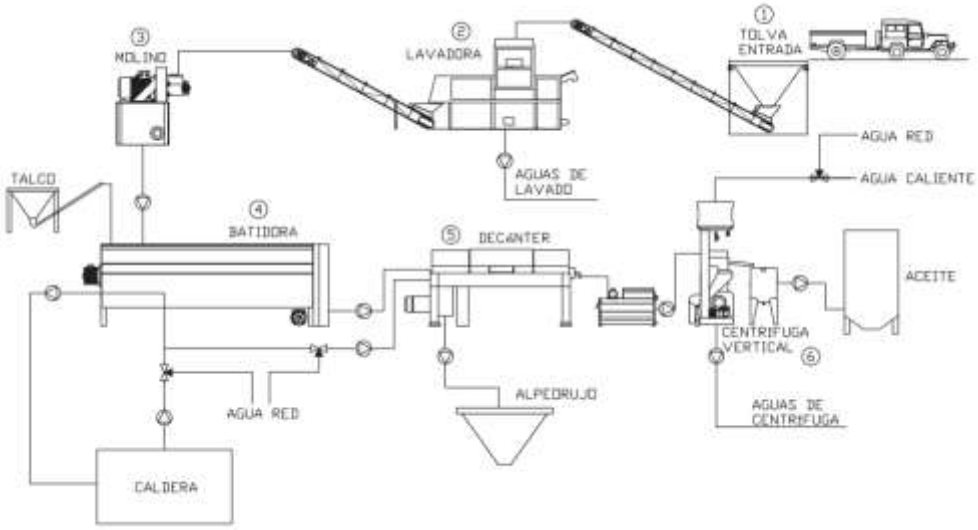


Figura 1.2 Proceso de elaboración de aceite de oliva virgen

A continuación la aceituna se carga en remolques para transportarla a las almazaras en el mismo día, de no hacerse en las 24 horas siguientes de su recogida, se produce un efecto indeseado en el fruto llamado atrojado, que es el responsable del aumento de la acidez del aceite. Almazara es un término que tiene su origen etimológico en el árabe al-mas'sara, que significa “extraer” o “exprimir”. Esta instalación consta de tres zonas básicas: la nave de recepción, la zona de elaboración y la bodega.

Los remolques llegan a la almazara donde se vierten las aceitunas en la tolva de recepción. Normalmente una almazara grande tiene más de una línea de recepción y existe un control de la calidad del producto de llegada. Automatizar el control de calidad sobre el estado del producto al llegar a la almazara es otro proyecto de investigación que daría para otro trabajo de este calibre. Según la calidad de la aceituna que entra en el proceso de obtención, saldrá un aceite de menor o mayor calidad. En ocasiones la recogida es muy sucia y por la tolva no solo entra fruto, sino que también se pueden ver hojas, tierra, piedras, etc., como incluso se aprecia en la imagen de abajo.

Figura 1.3 Tolva de entrada a una almazara

Es por ello que desde la cinta de entrada, en la mayoría de los casos, se pasa a un lavado, donde se elimina todo aquello que no sea fruto.

A continuación tiene lugar la molienda. Consiste en romper las paredes de las células de los tejidos vegetales de la aceituna liberando el aceite que contienen. El resultado es una pasta homogénea que contiene 4 elementos: aguas de vegetación, pulpa de aceituna, huesos y el aceite a extraer.

Seguidamente pasará por dos procesos de separación de elementos, lo que se conoce como el batido. Las batidoras se componen de varios cuerpos de acero inoxidable con un sistema de calefacción por agua caliente que se contiene en una capa que las rodea. Lo ideal es trabajar entre los 25 y los 30 °C.

Inicialmente se separará los sólidos de los líquidos, dejando por una parte las aguas de vegetación y el aceite y por otra parte la pulpa de aceituna y huesos de la misma, esto último se conoce como alpeorujo. Existen dos métodos para hacerlo: el tradicional se hace a partir del prensado en frío, que es una extracción mecánica realizada a baja temperatura. En la mayoría de los casos este método está ya en desuso. La segunda opción y más moderna es usar una centrífuga horizontal que separa los elementos por diferencia de densidad.

Dentro de la centrifugación existen dos sistemas: el sistema continúo de dos fases y sistema continúo de tres fases. Lo de sistema continuo se refiere a que la introducción de la masa obtenida en la molienda y la separación del sólido del líquido se ponen en marcha de forma continua.

- Sistema continúo de tres fases: se disponen dos salidas para líquidos y otra para sólidos. Por cada una salen separadas la fase más densa (aceite), la fase menos densa (orujo) y una fase intermedia (alpechín). Para este proceso tiene lugar la adición de agua entre 30 y 35 °C.
- Sistema continúo de dos fases: se disponen dos salidas, una para el aceite y otra para el alpechín y el orujo, que van mezclados.

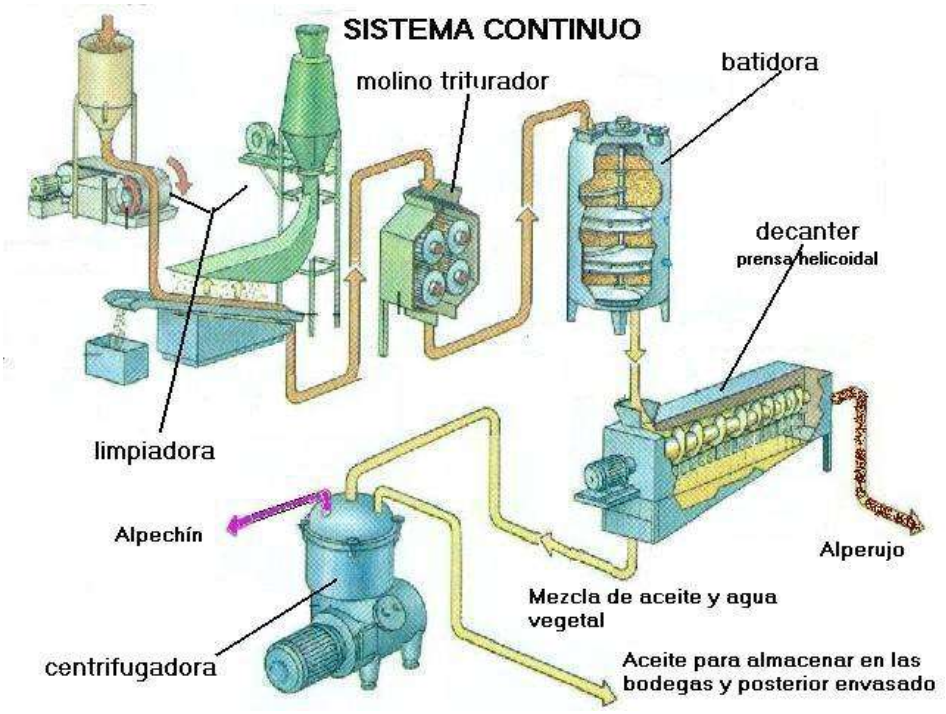


Figura 1.4 Sistema continuo de obtención de AOVE

El tercer paso de este proceso es una segunda separación del líquido resultante, ya que tendremos mezclados el aceite y las aguas de vegetación. Nuevamente existen dos opciones. Puede separarse los dos elementos por decantación, que consiste en separar por diferencia de densidades entre el aceite y el agua. Y en segundo lugar, el más efectivo y que garantiza más pureza, es usar una centrífuga vertical para separar perfectamente los dos elementos.

Figura 1.5 Zumo oleoso en el proceso de elaboración

Por último se almacena el producto en la bodega, un lugar con una atmósfera especial para la conservación y maduración del aceite, de manera que no se pierda calidad en el último paso.

Y bien, una vez que el proceso de obtención del producto está desarrollado y explicado, ¿qué papel juega la nariz electrónica dentro de él?

La nariz electrónica podría situarse en dos partes del proceso. Una de ellas sin duda sería la bodega. Donde el aceite ya está en reposo esperando a ser envasado y la nariz podría catalogar la calidad de la botella. Éste sería el lugar natural para el control de calidad de un producto acabado. Lo que emularía al control de calidad de la cata de aceite.

Sin embargo hay otra posibilidad de inserción de nuestro aparato en el proceso: A la salida de la centrifugadora vertical o el decanter. En este punto, la nariz podría detectar defectos en continuo en la calidad del aceite e incluso podrían intentar corregirse in situ. La validación de un concepto así tardará en confirmarse, ya que la nariz tendría hilar muy fino para alcanzar la precisión que se requiere.

Como conclusión del proceso actual de elaboración de AOVE, quiero destacar que aun contando cada vez más con técnicas modernas y avanzadas, es en general un proceso poco automatizado y que tiene un largo recorrido de robotización y transformación industrial. Desde que se inició este proyecto, ha estado siempre en mente el grado de automatización industrial al que se pretende llegar y lo que aportaría la nariz electrónica. Existen más proyectos de automatización dentro de las almazaras industriales, como, por ejemplo, el sistema NIR para la determinación del rendimiento graso del aceite estudiando el porcentaje de hueso y pulpa en el material sobrante, el orujo.

1.2.2. Calidad del AOVE

Hablando de rasgos globales, la calidad de un determinado producto es el conjunto de características propias que permiten bien distinguirlo de otros productos, bien apreciarlo como igual a otros productos.

Hablando concretamente del aceite, la calidad vendrá determinada por el zumo oleoso que se obtiene, según se ha explicado anteriormente, de una aceituna en perfectas condiciones, tanto en la recogida, elaboración y conservación. En definitiva que la manipulación y el tratamiento del fruto del olivo no altere la naturaleza química de sus componentes.

Un aceite de oliva virgen se define como aquel que ha sido obtenido del fruto del olivo únicamente por procedimientos mecánicos, en condiciones concretas y que no haya pasado por ningún otro tratamiento que no sea lavado, decantación, centrifugado y filtración.

Es muy importante diferenciar dos términos, calidad del aceite y tipo de aceite. El tipo de aceite se determina por las características particulares de cada variedad: Picual, Hojiblanca, Lechín de Sevilla, Morisca, Arbequina, Empeltre, Cornicabra, Blanqueta, etc. (Ver Figura 1.6) Las características de cada variedad son apreciables por las características organolépticas, que son, sabor, olor y color, y por la composición química. Naturalmente se pueden obtener dos aceites de la misma calidad aunque sean de distinto tipo.

Figura 1.6 Variedades de aceituna

Hay una gran cantidad de parámetros de calidad y pureza de los aceites de oliva. Los que se aplican usualmente al aceite de oliva virgen son la acidez, el grado de oxidación y las características sensoriales. Además de la detección de la calidad del aceite, es interesante medir la pureza, la cual indica posibles fraudes. Precisamente el fraude es otro de los objetivos que la nariz electrónica quiere abordar. Recientemente la controversia sobre este tema en la provincia de Jaén ha crecido considerablemente. El fraude consiste grosso modo en mezclar aceite de buena calidad con otro aceite de peor calidad o incluso aceite de orujo. Al tener composición química similar, es muy difícil notar el fraude que enturbia la industria.

Dentro de los parámetros de calidad y pureza, se distingue entre dos tipos, los parámetros químicos y los sensoriales. Los químicos son muy numerosos y complejos, por ello los citaremos y trataremos de sintetizar su valor.

El ya comentado grado de acidez, que se expresa en ácido oleico. El índice de peróxidos, que determina el estado de oxidación primaria de un aceite. Ceras, son compuestos que provienen de la esterificación de alcoholes alifáticos con ácidos grasos libres. Ácidos grasos saturados en posición 2 de los triglicéridos. Estigmastadieno, es un hidrocarburo esteroideo difícil de eliminar en el proceso de refinación. Diferencia entre ECN 42 (HPLC) y ECN 42 (cálculo teórico); el ‘Equivalent Carbon Number’ (ECN) se utiliza para detectar la presencia de pequeñas cantidades de aceites de semillas ricos en ácido linoleico. Absorbancia en el ultravioleta, esta

analítica se fundamenta en la medida espectrofotométrica ultravioleta del coeficiente de extinción a distintas longitudes de onda: 232 y 270 mm. Composición de ácidos grasos, permite distinguir información de la longitud y sobre la instauración de las cadenas hidrocarbonadas lo que va a permitir detectar mezclas de oliva con semillas.

En definitiva, hay muchos estudios químicos que se pueden aplicar en el aceite de oliva virgen y nuestro caso de estudio no contempla profundizar en ninguno de ellos. Solo es conveniente tener en cuenta que la determinación de la calidad y pureza de un producto no es trivial. (Morales, 2005)

Los parámetros sensoriales, es decir, las características organolépticas son las que se evalúan mediante el panel de cata, mediante un análisis sensorial de los aceites. Aquí es donde nace la subjetividad en la valoración ya que son personas las que realizan esta labor, aunque intenten ser jueces objetivos del análisis sensorial.

Figura 1.7 Guía Repsol de cata de Aceite de Oliva Virgen Extra

Llegados a este punto y retomando el tema de la controversia en el sector entre envasadoras y productoras, cito textualmente una notica del Diario Jaén de abril de 2018: ‘Los grandes envasadores insiste en la inseguridad’, titulaba el periódico. ‘Sus asociaciones afirman que no quieren quitar el panel de cata, pero dicen que hay que realizar mejoras’. La noticia proseguía del siguiente modo: ‘Anierac y Asoliva afirman que en ningún momento han pedido la desaparición del panel test como herramienta de calificación de los aceites de oliva, pero sí solicitan que se mejore su aplicación para que ofrezca garantías jurídicas a las empresas. Indican que esta herramienta para distinguir de forma comercial un aceite virgen, de un virgen extra o un lampante, basada en una cata de “sabor” ha demostrado en la práctica que tiene serios problemas por “variabilidad de resultados”, ya que se han encontrado numerosos

casos en los que una misma muestra ofrece clasificaciones muy diferentes según el panel de cata que lo valore'.

Otra noticia publicada en el mismo diario, titula así: 'Jaén abandera la protección de la cata para defender el buen aceite'. 'Jaén quiere reaccionar de forma firme a un documento que presentó la Asociación Española de la Industria y el Comercio Exportador del Aceite de Oliva (Asoliva) y la Asociación Nacional de Industriales Envasadores y Refinadores de Aceites Comestibles (Anierac) —avalado por unas 90 empresas— en el que se decía que el panel test, tal y como se concibe actualmente, genera inseguridad jurídica en las operaciones comerciales, sobre todo, en el extranjero. Además, dejaba ver que podía existir subjetividad en este método —hay estudios hasta que intentan diseñar una "nariz" electrónica que evite el componente humano—.'

Es un fragmento extraído de la noticia y la última oración entre guiones deja entrever la polémica que está latente en el mundo de las catas respecto a nuevas tecnologías que mejoren el análisis sensorial.

Una forma de objetivar la calidad, que ya consta, es el Índice Global de Calidad (IC). Para ello se utiliza una tabla de parámetros de calidad para aceites vírgenes propuesta por Gutiérrez y González Quijano y la siguiente fórmula de estandarización:

$$(I.C.) = IC\ 2{,}17 + 0{,}91\ \times PO - 0{,}81\ \times IA - 9{,}09\ \times\ K_{270} - 0{,}025\ \times IP$$

IC: Índice global de calidad

PO: Puntuación organoléptica

IA: Índice de Acidez

K_{270}: Transmisión al ultravioleta a 270 nm

IP: Índice de peróxidos

Figura 1.8 Envasadora de aceite de buena calidad

Tras discernir sobre los temas más relevantes de la calidad, se distingue tres tipos de aceite de oliva virgen según la misma.

- Aceite de Oliva Virgen Extra

Aceite cuyo ácido oleico no supera los 0,8 grs. por cada 100 grs., y las demás características cumplen lo establecido en esta denominación.

- Aceite de Oliva Virgen

Aceite cuyo ácido oleico no supera los 2 grs. por cada 100 grs., y las demás características cumplen lo establecido en esta denominación.

- Aceite de Oliva Lampante

Aceite de oliva virgen que tiene defecto o su ácido oleico es superior a 2grs. por cada 100 grs., y las demás características cumplen lo establecido en esta denominación. (Ver Esquema en Ilustración 1)

Todo lo que no cumpla la definición de aceite de oliva virgen, entra en la categoría de no virgen y la nariz electrónica tratará de distinguir los aceites vírgenes. Los aceites no vírgenes se entiende que no forman parte de la valoración subjetiva del panel de cata. (Jiménez Herrera, B. y Carpio Dueñas, A., 2008)

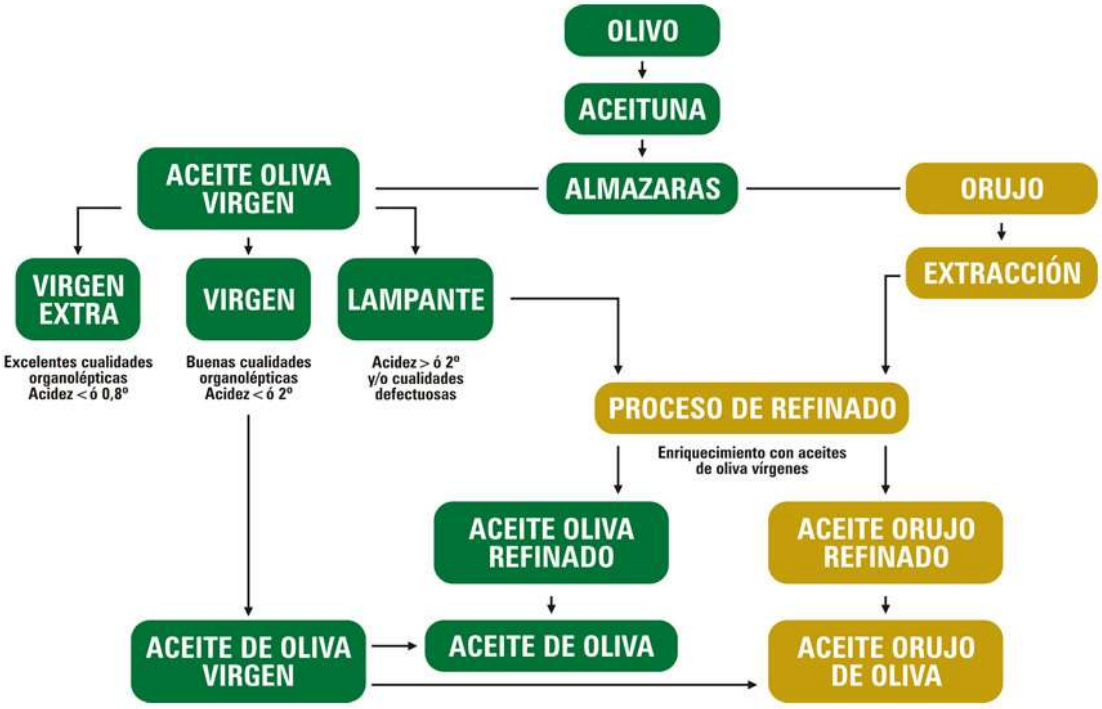


Ilustración 1 Esquema resumen del proceso de elaboración y calidad del aceite

1.2.3. La nariz electrónica y sus ventajas

Hay métodos en lo que se refiere al análisis de los aromas, entre los que destaca la cromatografía de gases. La cromatografía es eficaz en cuanto a resultados pero tiene ciertos inconvenientes:

- Coste alto de adquisición y mantenimiento.
- Se necesitan operarios experimentados.
- Obtención de resultados lento.
- Se requiere el uso de laboratorios.

La herramienta comúnmente utilizada en la industria alimentaria para testear el sabor o el aroma de productos es aún el olfato humano. Sin embargo, las deficiencias de este son notables. En primer lugar por el elevado coste de formar a un profesional en el análisis sensorial. Y en segundo lugar por la poca repetitividad de los resultados de estos análisis.

Por estos motivos nace la necesidad de crear una técnica que pueda suplir o ayudar al sentido del olfato y sea usado para obtener datos sensoriales en tiempo real a un coste reducido.

Nuestra nariz electrónica es una unidad constituida por una matriz de sensores químicos de gas y un sistema de reconocimiento de patrones, capaz de discriminar diferentes tipos de aceites. La nariz puede analizar el aroma, los componentes volátiles que desprende el aceite y determinar las huellas aromáticas características de cada uno de ellos.

Los sensores deben estar en una cámara estanca y sellada por donde se hace pasar un flujo de aire contenedor de los volátiles de la muestra. El aire se impulsa a través de la cámara con un sistema neumático compuesto por un motor, dos electroválvulas y tubos de conducción del flujo. Los componentes volátiles de la muestra se introducen mediante el sistema mecánico en la cámara de sensores. Estos reaccionan químicamente con cada tipo de aceite y la respuesta de los mismos se recoge y se procesa usando los mencionados patrones de reconocimiento.

El tipo de sensor más empleado en las narices electrónicas es el semiconductor de óxido metálico (MOS). La influencia de la temperatura y la húmeda en la cámara de sensores hay que tenerla muy en cuenta, y es por ello que además de sensores de gas, nuestra cámara cuenta con un sensor de temperatura y humedad relativa. (Boeker, Julio 2014)

A pesar de todas las explicaciones que se puedan dar, la nariz electrónica sigue siendo una máquina intrigante. Es difícil comprender que se trata de una reproducción del sentido biológico del olfato. Los receptores olfativos se sustituyen por los sensores de gas, el sistema nervioso por el microcontrolador y el software. La única deducción posible es que la nariz electrónica sea capaz de “oler”. ¿Pero es realmente así de fácil? Efectivamente no.

No se puede garantizar que una nariz electrónica puede tener fiabilidad completamente desde el primer momento. De hecho los sensores no huelen, los sensores miden. Pero antes de introducir cómo pasamos de la medición al olor que todos conocemos, ¿qué es el olor para el sentido humano?

Apenas se ha parado a pensar sobre esta cuestión y sus detalles en el mundo científico tecnológico. Desde una perspectiva general el sentido del olfato es un canal de información para volátiles del aire. Solo una fracción pequeña de todos los volátiles transporta información útil. Durante la evolución, el olfato se ha ido adaptando a la recepción de esos volátiles y a sus niveles de concentración. Un grupo de volátiles es considerado aroma, si nuestra conciencia registra su presencia y atributos y los identifica. Esto marca la frontera entre la medida de volátiles y la recepción del aroma.

Una notoria falacia en el mundo científico es esperar la información de un aroma con las señales resultantes en la nariz electrónica. No se puede esperar obtener un aroma en el sentido que nosotros recibimos el aroma, como una percepción humana. Lo que la nariz electrónica obtendrá serán trazos de señal de los aromas mezclados con la señal de volátiles mayoritarios que carecen de aroma. Por tanto, la nariz electrónica no simplemente indica la presencia de un aroma ya conocido, sino también información imperceptible sensorialmente por el olfato humano. (Majchrzaka, T.;Wojnowskia, W.; Dymerskia, J.; et al 2008)

1.2.4. Smart Sensors en la Industria 4.0

La Industria 4.0 (Ver Figura 1.9) es el ámbito natural de un proyecto como este. Se trata de una revolución industrial nueva, en la que la automatización y la robótica juegan un papel muy importante. El potencial que tienen los nuevos sensores y la inteligencia artificial reman a favor de esta corriente, mayormente relacionada con la ingeniería electrónica, pero que sin duda necesita la informática y la mecánica. Con la nariz electrónica se pretende entrar de lleno en esta revolución. Se pretende facilitar los procesos realizados por personas, mejorarlos y dar nuevas soluciones más eficientes. No se pretende sustituir completamente el factor humano por razones obvias. No es posible.

El sector industrial necesita una nueva etapa, marcada por la eficiencia energética y el IoT. El IoT, consiste en dotar de inteligencia a los objetos comunes de nuestro día a día, así como a máquinas y procesos industriales por completo. Para ello se necesita la conectividad a través de Internet, de tal manera que los objetos ‘puedan hablar’, se

puedan comunicar. La nariz electrónica tiene también como meta incluir esta tecnología y deja de ser una máquina industrial offline, sino online.

Figura 1.9 Industria 4.0

¿Y cómo ayuda en el proceso de obtención del aceite un aparato como éste? Para encontrar la utilidad real dentro de la industria, hay que basarse en un concepto muy novedoso y ventajoso. El Big Data, como se conoce en inglés, es una herramienta tan grande como potente. Grande por la idea básica per se de datos a gran escala y potente por la capacidad de sintetizar resultados concretos a partir de una gran masa de variables de entrada.

Traduciendo a nuestro caso de estudio, si hacemos pasar por la nariz electrónica un numeroso muestrario de aceite catalogado, podríamos encontrar un resultado concreto de clasificación del aceite. Los primeros resultados conducirían a clasificaciones simples como, virgen extra o virgen. Sin embargo un uso de la herramienta comentada anteriormente daría lugar a un fiable discriminador de la calidad del aceite. (Mildner-Szkudlarz, octubre 2008)

En definitiva, la industria de la almazara en la provincia de Jaén, así como en el resto del mundo, está sufriendo una transformación que viene a mejorar y dar una nueva visión al sector.

1.3. Objetivos

1.3.1. Objetivo general

El objetivo general del Trabajo Fin de Grado es obtener un Smart Sensor, la nariz electrónica, para la discriminación de la calidad y tipos de aceite a partir del análisis de sus componentes volátiles y que pueda ser integrado en la línea de producción.

- Obtención de un prototipo optimizado

Uno de los factores clave de la nariz electrónica es su introducción en el mercado de la obtención de aceite. Para ello, el producto que se obtenga debe tener un precio competitivo y al alcance de todos los bolsillos, capaz de ser fiable y asequible. No quiere esto decir que se escatime en la utilización de materiales, ni en componentes electrónicos de la máxima calidad, sino que se trata de democratizar las nuevas tecnologías.

- Discriminación de la calidad y tipos de aceite a partir del análisis de sus componentes volátiles

Es importante destacar que, la nariz electrónica mide los componentes volátiles que desprende el aceite en unas determinadas condiciones. Es imposible sustituir el factor humano completamente porque las personas sentimos con el gusto y el olfato a la hora de probar un aceite. La nariz electrónica solo medirá aquellos elementos que se puedan volatizar en el aceite y la validez de la huella caracterizada solo dependerá del cambio en la medida de los sensores para distintos tipos de aceite

- Entrada de nuestra unidad en el proceso de obtención del aceite en las almazaras

No solo se pretende que la nariz funcione en laboratorio, o en proyectos de investigación, sino también en el propio proceso de la industria. Donde se considera que es realmente útil ya que permite valorar el aroma de la muestra de aceite en tiempo real. En el futuro una posibilidad será retroalimentar la información de los sensores en el proceso de obtención, con el objetivo de cambiar parámetros influyentes en la calidad del aceite.

1.3.2. Objetivo específicos

- Estudiar y seleccionar las herramientas software que se utilizarán para desarrollar la aplicación.

Todo el software creado para este trabajo es hecho de principio a fin exclusivamente para el proyecto de la nariz electrónica y, antes de empezar a crear, el objetivo principal era aprender a utilizar las herramientas citadas anteriormente.

- Diseñar la estructura software de la nariz electrónica.

Desde la obtención de datos a través de los sensores, hasta su representación y todas las funcionalidades intermedias. El software de desarrolla en Java, en el entorno de programación NetBeans IDE 8.0.2 y como soporte, el conocido y novedoso micro controlador Raspberry Pi 3 B. Esta configuración permite entrar en el mundo del IoT gracias a la conectividad WiFi y Bluetooth de Raspberry. En Java se trabaja con la programación basada en agentes.

- Implementar la aplicación software diseñada.

Para el correcto funcionamiento de la unidad, el código se trata de hacerlo lo más ordenado posible y con la suficiente robustez de un producto industrial. No es un proyecto que se inicie solo para un trabajo final de grado, sino que se espera que se recoja el testigo y se evolucione cada vez más hacia un acabado mejor en todos los sentidos. Creando nuevas versiones y cambiando lo necesario en el software. Para ello, es primordial que la programación sea comprensible para futuras personas encargadas de ella.

- Validar la aplicación desarrollada con un caso de estudio.

El periodo de validez y testeo de la nariz electrónica es un paso fundamental para establecer si nuestra huella del aceite es un indicador fiable o no.

1.4. Contexto

Figura 1.10 Interacción GRAV e ISR

Ya en la introducción se ha mencionado brevemente de donde surge la idea de desarrollar una nariz electrónica con fines comerciales. En el contexto del Grupo de investigación de Robótica, Automática y Visión por computador, GRAV, son muchos los proyectos que tienen que ver con el sector agro. Todos ellos basados en la automatización de procesos para la mejora continua de la industria en nuestra provincia. En el grupo no solo se trabaja el sector agro, aceite en definitiva, sino que además hay una diversidad de proyectos nutrida por las necesidades de las empresas. Esas necesidades suelen ser problemas de solución compleja, por ello tiene sentido que el conocimiento de la Universidad sea la principal fuente de transferencia en el desarrollo de estas soluciones.

ISR es una spin-off de la Universidad de Jaén que surge del grupo de investigación GRAV y nace con la finalidad de abarcar proyectos cada vez más grandes. Se considera una pequeña empresa de base tecnológica que está dando sus primeros pasos en el sector y mantiene estrecha relación con el GRAV. La empresa cuenta con una amplia experiencia en el desarrollo de nuevas soluciones tecnológicas y de productos basados en la integración sensorial y en la automatización avanzada, ya sean para la industria, en general, como para el sector AGRO.

El caso del estudio de la nariz electrónica no es ni mucho menos nuevo dentro de las líneas de investigación del GRAV. Viene de lejos el estudio sobre la posibilidad de caracterizar la calidad del aceite de oliva a partir de sus componentes volátiles, es

decir, analizar sensorialmente las propiedades organolépticas con sensores de gas. El estudio previo realizado no es más que la antesala a todo lo que posteriormente se ha seguido estudiando y poniendo en práctica. Bajo este marco de investigaciones sobre la materia, puede parecer que empezar el proyecto fue relativamente sencillo, sin embargo la complejidad residía en contrastar toda esa información disponible con las posibilidades de fabricación reales, eficaces y optimizadas.

Figura 1.11 Sector Agro en ISR

En el proceso previo de investigación del personal docente, también se han realizado ensayos con muestras de aceite catalogadas en panel de cata y con una nariz electrónica comercial. Los resultados que arroja esta nariz electrónica nos da pistas de cómo debe comportarse la nuestra, el rendimiento y la fiabilidad que podemos esperar. La gran diferencia de la nariz comercial disponible en el grupo y la nariz a desarrollar consiste en la tecnología y en el precio. No quiere decir que una tecnología sea peor que otra, sino que se busca un producto optimizado que influya precisamente en la reducción de costes para conseguir un precio competitivo.

Para contextualizar el trabajo global realizado en cada parte de la nariz electrónica del GRAV, se verá brevemente el desarrollo de electrónica hardware y mecánica.

En electrónica hardware, utilizar Raspberry, comunicación I2C y hasta la elección de cada sensor ha influido en la programación y el diseño del software.

Se han realizado tres placas PCB, (Hernández Cledera, M., 2018):

- Placa de sensores (Ver Figura 2.1)

Se trata de una tarjeta de adquisición de datos, la cual también incluye un sistema de acondicionamiento de la señal específica para cada sensor. Basa su funcionamiento en el protocolo de comunicación I2C.

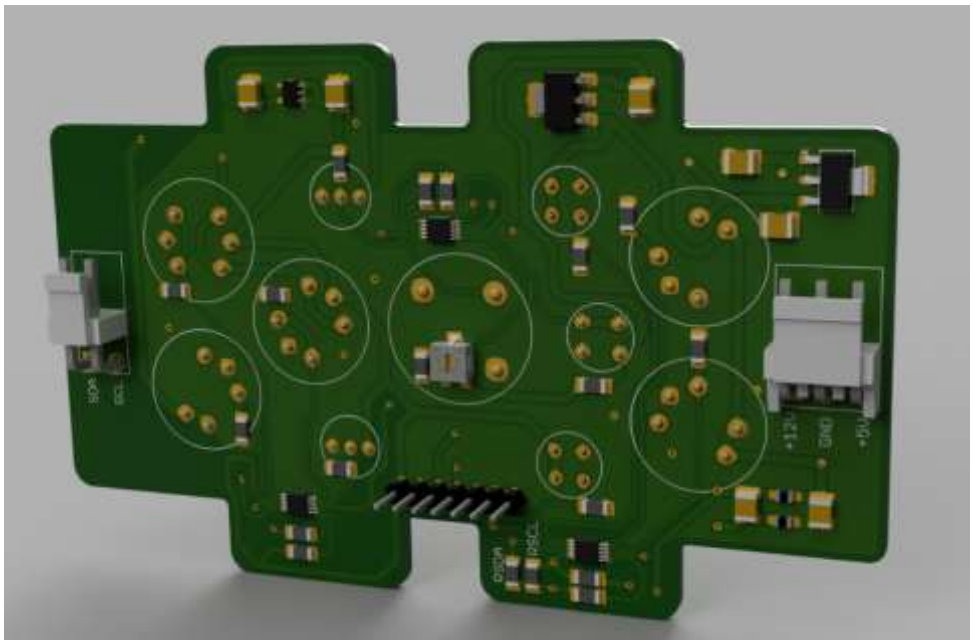


Figura 1.12 Placa de sensores

- Placa bornero de Raspberry (Ver Figura 2.2)

Es un circuito modular que actúa como pasarela entre Raspberry y periféricos para adecuar su funcionamiento al sistema embebido diseñado para la nariz electrónica.

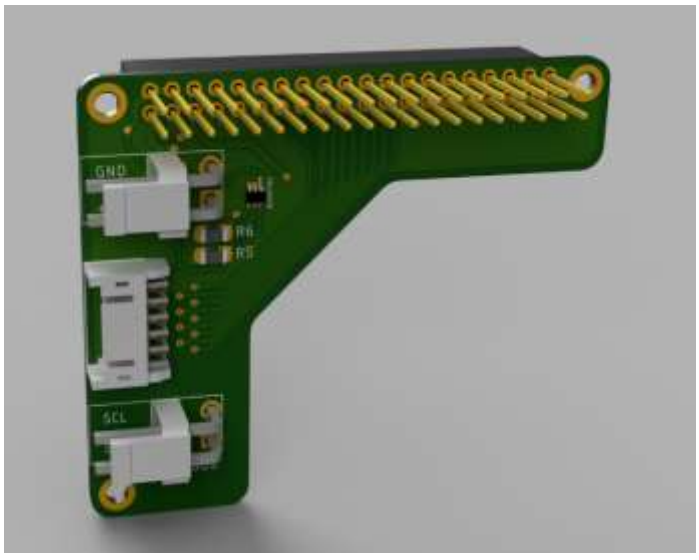


Figura 1.13 Placa bornero Raspberry

- Placa de relés (Ver Figura 2.3)

Es un control de cargas basado en el protocolo de comunicación I2C. Será la tarjeta encargada de controlar actuadores que requieran de altas capacidades energéticas, aislándose del control a bajo nivel de la Raspberry.

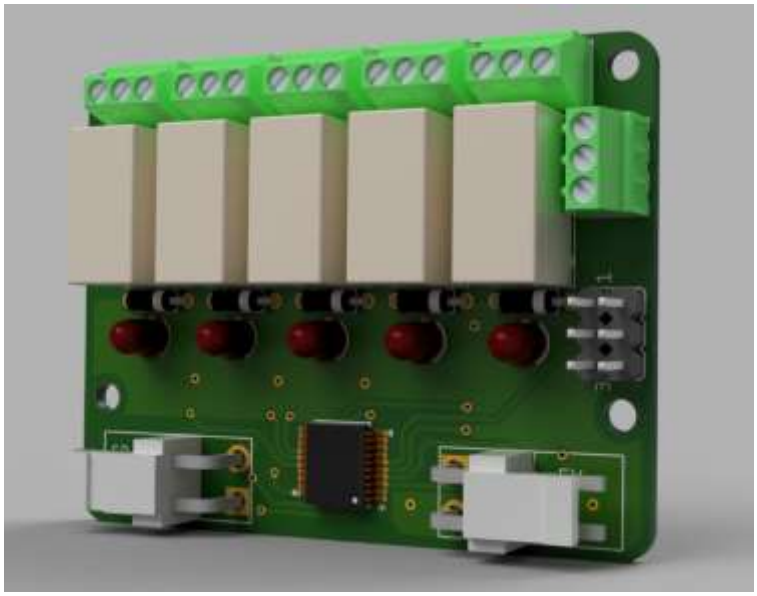


Figura 1.14 Placa de relés

En la parte mecánica se ha diseñado toda la estructura que da cabida a la electrónica, desde la caja que guarda al dispositivo, la cámara de sensores por dónde se hace pasar el flujo de volátiles, el circuito neumático y la elección de los elementos neumáticos, simulación de flujos del aire y el mecanizado de todas las piezas necesarias. (Gómez García, J., 2018)

Figura 1.15 Cámara de sensores

1.5. Estado del arte

Con el objetivo de situar el proyecto en un marco histórico tecnológico, pondremos en antecedentes la evolución del concepto de la nariz electrónica hasta nuestros días.

Los orígenes se remontan a la mitad de siglo pasado, donde se empezaron a construir dispositivos con un solo sensor de gas, los cuales no se podían considerar si quiera nariz electrónica. En los años ochenta surgen grupos de investigación que publican en revistas artículos sobre narices electrónicas, orientando dichos artículos a la compresión del olfato biológico a través de sensores semiconductores de óxido metálico. Es a finales de siglo pasado cuando realmente aparece el concepto de nariz electrónica como sistema inteligente. (Ver Figura 1.12)

Una de las primeras definiciones y más populares es la de Gardner y Barlett (Gardner y Barlett., 1999): "*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*".

Todos estos avances han dado lugar a la comercialización de narices electrónicas, ya sea para usos de investigación, de seguridad o sanitarios, (Moreno, I., julio 2009).

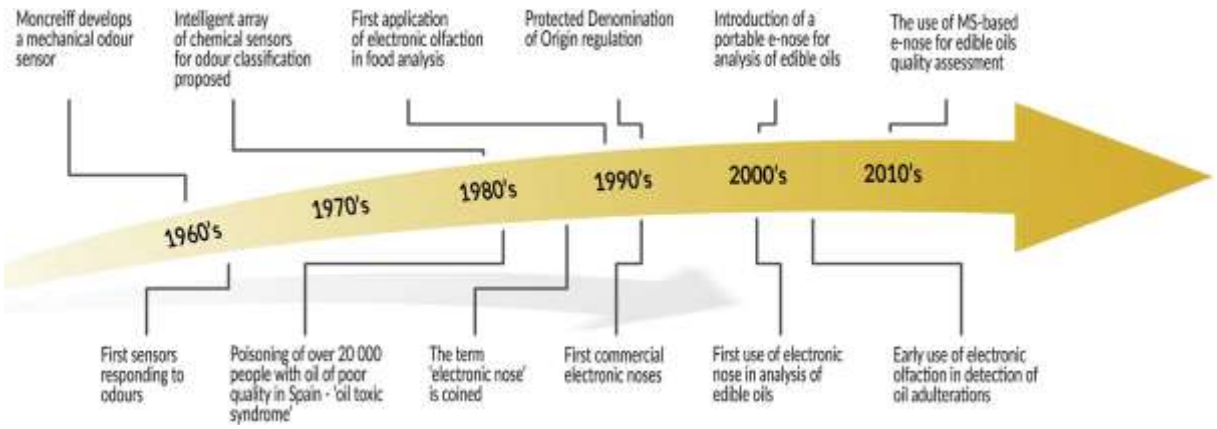


Figura 1.16 Evolución histórica del estado del arte

Las narices electrónicas fueron desarrolladas en un comienzo para reemplazar a los paneles de expertos en la clasificación de aromas, dado su alto coste, complicado

transporte, su subjetividad por verse afectados por cansancio, estado anímico, clima y factores externos. Distintas empresas de varios países han apostado por la fabricación de narices electrónicas:

Empresa	Producto	Características
Airsense Anális GMBH	PEN3, i-PEN	Pueden identificar un gas o mezcla de gases.
AlphaMos	Prometheus, Fox y Kronos	Kronos ofrece un análisis cualitativo y cuantitativo de las muestras.
Chemsensing INC		Detección de tumores de pulmón.
EADS-RST Rostock System-Technik GmbH		Detección temprana de fuego en áreas peligrosamente explosivas.
Environics Industry Oy	M90-DI-C, ChemPro 100	La ChemPro 100 detecta agentes químicos de guerra y componentes o materiales industriales tóxicos.
Electronic Sensor Technology	*Model 4200 zNose, Model 7100 Benchtop zNose.	Identifican trazos de componentes orgánicos, biológicos y químicos.
INFICON	Smart Chemical Identification System	Analizador de compuestos orgánicos volátiles del medio ambiente.
Scensive Technologies	Bloodhound ST214	Es capaz de cubrir todo tipo de olores.
Technobiochip	**Libra Nose	Es compacta y fácil de utilizar.

Figura 1.17 Empresas que fabrican narices electrónicas

Figura 1.18 Portable Electronic Nose de Airsense

2. METODOLOGÍA

En este capítulo se explicará con detalle todos los aspectos del procedimiento seguido para el desarrollo del software de la nariz electrónica. La parte software ha estado muy ligada durante todo el proyecto a la parte hardware realizada por uno de los compañeros. El trabajo en equipo ha sido fundamental, desde el diseño inicial de cada parte hasta la consecución final de un prototipo. Ha consistido en un desarrollo en paralelo de ambas partes, además del desarrollo mecánico de la nariz electrónica.

Lo que se hizo en primer lugar, aunque no se recoja en detalle en este trabajo, fue la elección de sensores de la nariz electrónica, teniendo en cuenta el convertidor que se usaría para la obtención de los datos. El convertidor tiene sus propias librerías de Adafruit, que posteriormente han sido realizadas para la programación en Java. La selección de sensores de gas fue un proceso largo, sobre todo por la amplitud del mercado y las posibilidades de varias tecnologías existentes. Como el objetivo de la nariz electrónica era hacer un producto optimizado, la tecnología del sensor no podía encarecerla demasiado. Los elegidos son los sensores de gas tipo MOS, precisos y eficaces, pero a la vez un precio razonable con el que poder competir.

En segundo lugar y partir de este punto de partida se inició el progreso continuo de cada una de las patas del proyecto. A continuación comenzaremos a explicar cómo se ha avanzado en el apartado del software, hasta el día de finalización del proyecto.

Para poder cumplir con los plazos previstos, la metodología fue ordenada desde cuestiones básicas que hubo que resolver, hasta el último momento de diseño del software. El orden de los puntos tratados en el índice, también coincide cronológicamente con la realización de cada uno de ellos. No por otro motivo especial, sino que para llevar a cabo la siguiente tarea era necesaria la anterior.

2.1. Plataforma de desarrollo

2.1.1. Raspberry Pi

La plataforma de desarrollo Raspberry puede ser aún desconocida para muchos, ya que su lanzamiento es muy reciente. Sin embargo estos aparatos son muy conocidos en el contexto de la ingeniería electrónica, ya que resultan muy útiles para los nuevos avances y para desarrollar soluciones a las nuevas necesidades que se

dan en el sector industrial. Pero, ¿qué es realmente Raspberry Pi? Es un ordenador pequeño y asequible, de bajo consumo, que se puede usar como soporte para programación de alto nivel. Generalmente este tipo de ordenadores tienen sistemas operativos basados en Linux y tienen relación estrecha con el Open Hardware.

Fue desarrollado en Reino Unido por la Fundación Raspberry Pi en 2011, con un objetivo académico y de acercar a todos los públicos el uso de un ordenador. Su comercialización comenzó al año siguiente y se vendieron millones de unidades de su primera versión. El concepto es el de un PC descubierto totalmente, solo tiene su placa tipo ordenador con los elementos principales e imprescindibles. Una de las características más importantes del concepto es la miniatura que se ha conseguido, que con la capacidad de computación que tiene es capaz de abordar proyectos grandes. Ni que decir tiene que la potencia de Raspberry no es equiparable a la que tienen los ordenadores que usamos día a día.

De forma resumida, todas las versiones cuentan con un procesador, un chip gráfico y una memoria RAM. Además Raspberry incorpora funciones propias de la electrónica tales como pines GPIO (General Purpose Input/Output) y de comunicación entre dispositivos como UART (Universal Asynchronous Receiver-Transmitter), SPI (Serial Peripheral Interface) y I2C (Inter-Integrated Circuit). Estas funciones permiten emplear este mini-ordenador en proyectos de electrónica y robótica que interactúen con sensores y actuadores.

Esta versatilidad del dispositivo contribuye enormemente a la popularidad en la comunidad electrónica. El éxito de Raspberry ha garantizado ir un paso más allá en el abanico de posibilidades. Los usos son casi infinitos, tantos como se puedan imaginar.

Como toda marca comercial, Raspberry tiene sus propias versiones y modelos. Todas ellas se diferencias de las otras, normalmente en mejoras conforme sube el modelo y la versión del mini PC.

En el caso de la nariz electrónica, la versión y el modelo que utilizamos es la Raspberry Pi 3 Model B (Ver Figura 2.4). Esta placa cuenta en un espacio muy reducido con una serie de características que la hacen tremendamente interesante. La Raspberry Pi 3 Model B es el modelo más temprano de la tercera generación. Reemplaza al modelo antiguo Raspberry Pi 2 Model B y se lanzó en Febrero de 2016.

Figura 2.1 Raspberry Pi 3 Model B

Las características principales del modelo son:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Bajo Consumo (BLE) en la placa
- 100 Base Ethernet
- 40 pines GPIO
- 4 puertos USB 2
- Jack de audio 3.5mm y puerto de video compuesto
- Full size HDMI
- Puerto CSI camera para conectar a Raspberry Pi camera
- Puerto DSI para conectar Raspberry Pi a una pantalla táctil
- Puerto Micro SD para cargar el Sistema operativo y guardar datos
- Micro USB con alimentación de hasta 2.5A

Las características más interesantes y las que queremos recalcar, sobre todo a fin de justificar también la elección de esta plataforma, son la conectividad del dispositivo con otros dispositivos a través de internet y bluetooth. Además de cuatro puertos USB disponibles para periféricos y la salida HDMI.

Figura 2.2 Ícono Raspberry e ícono Raspbian

La Raspberry Pi a nivel de usuario puede funcionar muy bien con periféricos acoplados a sus puertos. Por ejemplo, monitor, cable HDMI, teclado USB, ratón USB, fuente de alimentación, etc. Además necesita la tarjeta SD que contenga el sistema operativo, normalmente Raspbian.

Raspbian es el sistema operativo recomendado para Raspberry Pi (al estar optimizado para su hardware) y se basa en una distribución de GNU/Linux llamada Debian. En el caso de este trabajo, se requería como en cualquier proyecto con Raspberry su puesta a punto. Tanto cargar el sistema operativo en la SD como configurarla.

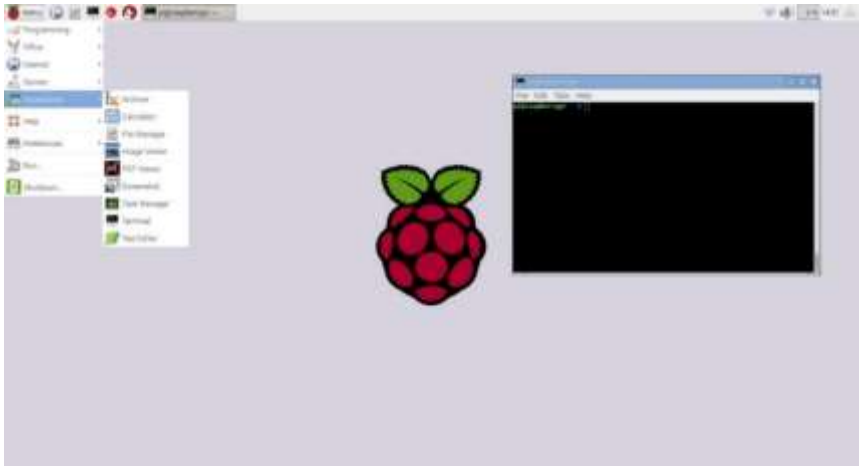


Figura 2.3 Entorno Raspbian

A la hora del desarrollo de este proyecto, se necesitaba no solo utilizar la Raspberry Pi a modo de usuario, sino que el enlace con el lenguaje de programación Java y la potencia de cálculo requerida nos obligaba a utilizar un paso más allá los recursos de la placa. En los siguientes subapartados del apartado de Plataformas de desarrollo explicaremos los recursos necesarios para avanzar en la programación del software de la nariz electrónica.

2.1.2. JAVA

Java es un lenguaje de programación orientado a objetos creado en 1991 y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Recientemente ha sido adquirida por Oracle en 2010.

La filosofía de Java es que los programadores escriban código solo una vez y puedan ejecutarlo en cualquier dispositivo. Esto es posible ya que Java cuenta con una máquina virtual llamada, Java Virtual Machine (JVM), que brinda la portabilidad al lenguaje. JDK posee un compilador que coge el código creado en Java y valida la sintaxis. Si hubiera algún error en el código el compilador nos mostrará un mensaje y si no hay error, el compilador creará un archivo con código byte .class, que será ejecutado por el JVM.

Al ser un lenguaje derivado de C tiene una sintaxis muy similar. En cuanto a estructura, el lenguaje Java comienza con paquetes, dentro de los cuales están todas las clases y dentro de las clases se crean los métodos, constantes, variables, etc.

Java es un lenguaje de programación que tiene múltiples ventajas. Entre otras, es uno de los lenguajes más utilizados en la actualidad, Android tiene una base importante de Java, existen un gran soporte, documentación y ayuda en la comunidad de internet. Java además cuenta con una extensa serie de librerías que multiplican las funcionalidades de una aplicación. Su máquina virtual es segura y está protegida ante infiltraciones y por último, Java está diseñado para crear software altamente robusto y fiable.

Por todas estas razones, se tomó la decisión de programar en Java la nariz electrónica y además, por la facilidad de combinar el soporte electrónico Raspberry de forma remota y dicho lenguaje. Esta combinación consiste en lanzar remotamente el programa en la Raspberry, pero crearlo en un ordenador personal. El ordenador portátil actúa solo como elemento de programación, pero una vez que el código ya está creado o se quiere probar, corre en el mini PC. Lo cual da autonomía a la nariz electrónica y libera potencia de cálculo en la programación de una aplicación tan grande.

La plataforma del entorno Java que permite usar la funcionalidad remota explicada es JRE (Java Runtime Enviroment). Este entorno de ejecución no viene instalado por defecto en Microsoft. Más adelante describiremos las entrañas de la arquitectura creada para el desarrollo del software entre todos los componentes del sistema.

Como en todos los lenguajes, Java tiene sus propias expresiones, ese conjunto de elementos que permiten crear el código. También son llamados 'tokens'; son el elemento más pequeño de un programa y son interpretados por el compilador. En Java podemos distinguir cinco tipos de 'tokens'.

- **Identificadores**: Representaciones que se les da a los nombres que se asignan a las variables, clases, paquetes, métodos y constantes para que el compilador pueda procesarlos.
- **Palabras clave**: Son palabras o identificadores especialmente reservados para desempeñar una tarea concreta. Se usan en casos específicos.

Éstas son: abstract, boolean, break, byte, case, cacth, char, class, continue, default, do, double, else, extends, false, final, finally, float, for, if, implements, import, intstanceof, int, interface, long, native, new, null, package, private, protected, public, return, short, static, super, switch, synchronized, this, throw, throws, transient, true, try, void, volatile y while.

- **Literales y constantes**: Sirven para asignar valores a una variable. También puede ser una variable de tipo numérico.
- **Operadores**: Nos indican una evaluación que se aplica a un objeto o un dato. Hay muchos tipos de operadores. Se recogen en la siguiente tabla:

Operador	Descripción
-	Cambio de signo
!	Operador NOT
~	Complemento a 1

Tabla 2.1 Operadores unarios

Operadores	Descripción
+ - * / %	Operadores aritméticos
== != < > <= >=	Operadores relacionales
&& || ^	Operadores booleanos
^ << >> >>>	Operadores a nivel de bit
+	Concatenación de cadenas

Tabla 2.2 Operadores binarios

- **Separadores**: Indican al compilador de Java donde se ubican los elementos del código y además extraoficialmente ordenan el código.

  Éstos son: { } , : ;.

La siguiente tabla muestra los tipos de datos que se pueden utilizar en Java y su posible conversión a otro tipo de dato, lo que resulta muy útil en algunos casos. Al ser un lenguaje derivado de C, naturalmente los tipos de datos son iguales o parecidos.

Tipo origen	Tipo destino
byte	double, float, long, int, char, short
short	double, float, long, int
char	double, float, long, int
int	double, float, long
long	double, float
float	double

Tabla 2.3 Tipos de datos

Después de este resumen de conceptos básicos de Java, dedicaremos los dos siguientes apartados a tratar en detalle dos bases de la programación de la aplicación de la nariz electrónica. El primero de ellos la programación basada en agentes o multihilos y el segundo las librerías gráficas para la representación de datos.

2.1.2.1. Programación basada en agentes

La programación en Java basada en agentes era un concepto totalmente nuevo para mí. Se trata de una ejecución múltiple de hilos en paralelo. Cada agente ejecuta al mismo tiempo un hilo con el código que tiene programado. Digamos que

simultáneamente están corriendo todos los agentes a la vez y solo se finalizará su ejecución al salir de la aplicación. Los agentes tienen la capacidad de comunicarse entre sí para enviarse mensajes y transferirse datos. Esta comunicación actúa de forma parecida a una interrupción, es decir, un primer agente se comunica con un segundo agente enviando un mensaje solicitando un determinado dato. En ese momento la ejecución en el hilo del primer agente se para y el segundo agente, al recibir la comunicación del primero, contesta con otro mensaje al primer agente y a su vez, envía el dato.

La razón por la que se eligió esta estrategia de programación fue la potencia computacional y la velocidad de ejecución que tienen los agentes frente a la programación convencional. Los agentes permiten tener varias funciones ejecutándose a la vez, lo cual ahorra mucho tiempo en el ciclo de un programa. También permiten con facilidad el paso de variables de una clase a otra y la autonomía de cada clase sin depender de ninguna otra.

Cada agente puede tener asociado otras clases que no sean agentes, por ejemplo, una clase con una pantalla o una simple función. Las clases asociadas solo se pueden comunicar con otras clases a través de sus agentes. En algunos casos esto puede ser un impedimento si queremos compartir datos entre clases asociadas que no sean agentes, porque siempre debemos pasar los datos por sus respectivos agentes para que se puedan comunicar.

Para crear la configuración de agentes son necesarias distintas clases: cinco de ellas en el paquete fuente ‘es.agent.core’ y dos en el paquete fuente ‘es.controller’. Las cinco clases de ‘es.agent.core’ son configuración inicial de la programación basada en agentes. Estos paquetes vienen dados en librerías y solo es necesaria su instalación. Las dos clases del ‘es.controller’ son el Main.java y el Agents.java, las cuales sí que podríamos editarlas si nos conviene.

La clase Main.java solo inicializa la aplicación de agentes, no tiene otra función. La clase Agents.java sí tiene funciones más cercanas al programador. Es en esta clase donde podemos declarar e instanciar nuevos agentes.

Ilustración 2 Flujograma Agents.java

Las clases del paquete ‘es.controller’ son las que han sido programadas con el diseño específico para la nariz electrónica. En este caso hay cinco clases agentes que controlan todo el programa, aunque una de ellas es la que principalmente guía el flujo de la programación.

El diseño de los agentes es la primera tarea que se llevó a cabo antes de empezar a programar como tal y no es una tarea sencilla, ya que es en el diseño cuando realmente se piensa en las funciones de la aplicación y cómo distribuirlas en el programa. Además también se diseñaron las interfaces de usuario y sus características. El primer diseño corresponde a la Figura 2.7 que se muestra a continuación:

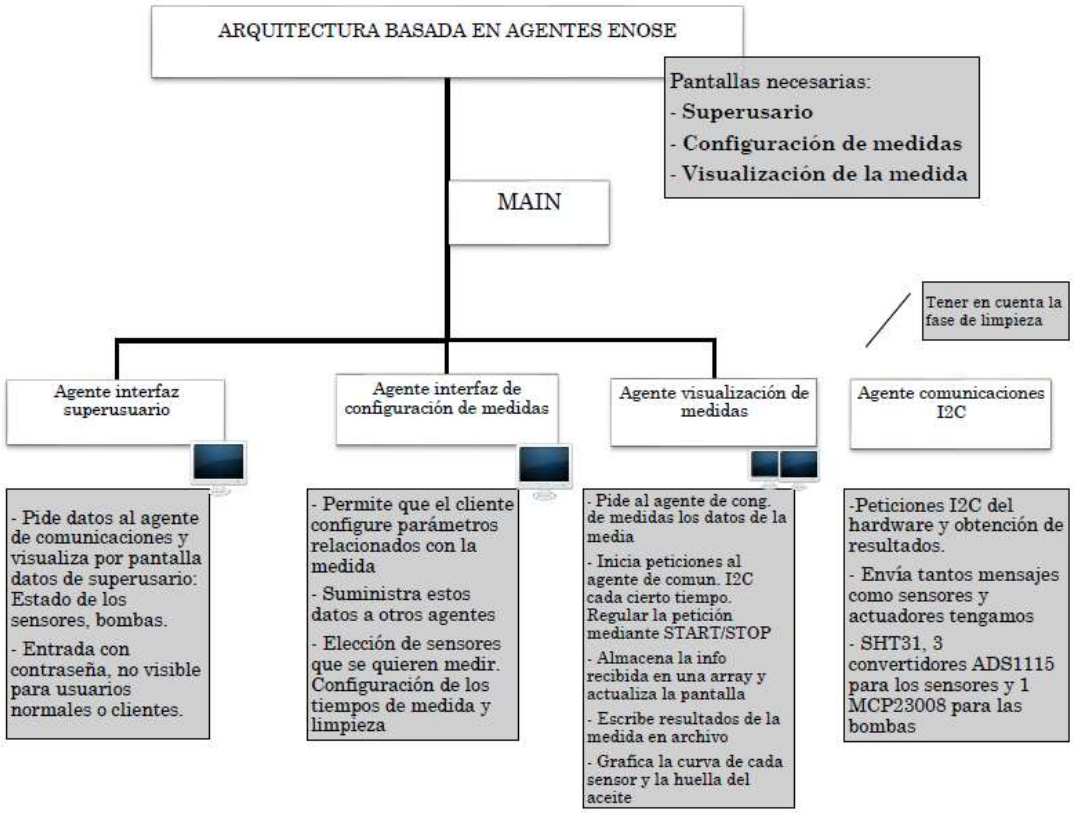


Figura 2.4 Primer diseño con la filosofía de agentes Java

Aunque éste fue un primer diseño de la aplicación, el resultado final no ha sufrido severas modificaciones. Sí ligeros cambios en el número de agentes, uno más, y en la estrategia de afrontar todo lo que había que abordar en la interacción con la electrónica hardware. Todo ello quedará recogido más adelante en otro punto.

Un agente se divide siempre en dos métodos:

- **toInit**

Método que se ejecuta solo una vez en la clase del agente. Se utiliza para ejecutar un proceso que se tenga que hacer al principio en la aplicación.

- **toProcess**

Método que se ejecuta de una forma especial. Está controlado por ‘switch’ e implementa en cada ‘case’ la recepción de un mensaje y lo que se debe ejecutar a continuación después de recibir ese mensaje. Se pueden implementar tantos

‘cases’ como sean necesarios y los mensajes se inicializan globales de tipo ‘public final static String’ para que sean reconocidos. Dentro de cada ‘case’ se pueden hacer múltiples tareas, desde enviar un mensaje a otro agente, hasta llamar a una función de clase asociada al agente.

2.1.2.2. Librerías Java

En Java y en varios lenguajes de programación más, existe el concepto de librerías. Una librería en Java se puede entender como un conjunto de clases, que poseen una serie de métodos y atributos. Lo realmente interesante de estas librerías para Java es que facilitan muchas operaciones. De una forma más completa, las librerías en Java nos permiten reutilizar código, es decir que podemos hacer uso de los métodos, clases y atributos que componen la librería evitando así tener que implementar nosotros mismos esas funcionalidades.

Hacer uso de librerías tanto propias como externas, o librerías propias de Java, es bastante fácil. Básicamente lo único que debemos saber es que para importar librerías en Java se usa la palabra clave ‘import’ seguido de la "ruta" del paquete o clase que deseamos agregar al proyecto. Cabe resaltar que el ‘import’ permite agregar a nuestro proyecto una o varias clases (paquete) según lo necesitemos.

Aunque el uso de distintas librerías programando en Java es muy frecuente, destacaremos dos de ellas que dadas las condiciones del diseño de nuestra aplicación se hacen obligatorias.

La primera de ellas es Pi4J, una librería I/O de Java especial para Raspberry Pi. Da soporte a una API de entradas/salidas orientada a objetos e implementa para programadores de Java todas las capacidades con las que poder acceder a la plataforma Raspberry Pi. Trata de ser un puente entre Java y las librerías nativas de Raspbian, sistema operativo de Raspberry. Tiene también soporte para varias versiones de Raspberry, incluyendo la Pi 3 Model B, y aunque es una librería con años de estabilidad, sufre continuamente cambios para la mejora de sus capacidades. En concreto lo que nos permite esta librería es habilitar la comunicación I2C entre Raspberry y Java.

Como se aprecia en la Figura 2.8, el modelo de Raspberry que se ha usado tiene una cabecera de 40 pines de expansión, de los cuales 28 son pines GPIO. Ya sean para comunicación I2C, SPI o comunicación Serie, los pines GPIO sirven para establecer comunicación con el exterior a la controladora.

Figura 2.5 Expansión pines Raspberry Pi 3 Model B

La Figura 2.9 que se muestra más abajo es un gráfico resumido de todas las funciones que tiene la cabecera de pines del modelo de Raspberry. Tiene 12 pines de tierra y alimentación a varias tensiones y los 28 pines GPIO ya mencionados.

Para el desarrollo de la nariz electrónica, la comunicación escogida fue I2C. Los pines 3 y 5 del 'header' corresponden con los canales de comunicación I2C SDA y SCL respectivamente. Es en estos pines donde la parte hardware ha realizado circuitos de conexión con el bus de datos I2C de la placa de sensores que recogía todos los datos de los sensores. Más adelante se explicará en detalle cada elemento del bus.

Por otra parte, los componentes de la nariz electrónica también necesitan su alimentación concreta. En el caso de los sensores, hay algunos de ellos que se alimentan a 5V y otros a 12V, pero en ningún caso se alimentaban de los pines de la Raspberry. La placa de sensores lleva su alimentación separada con una fuente de alimentación de 12V. Lo que sí se alimenta a partir de los pines de 5V de la Raspberry es la pantalla de la nariz electrónica.

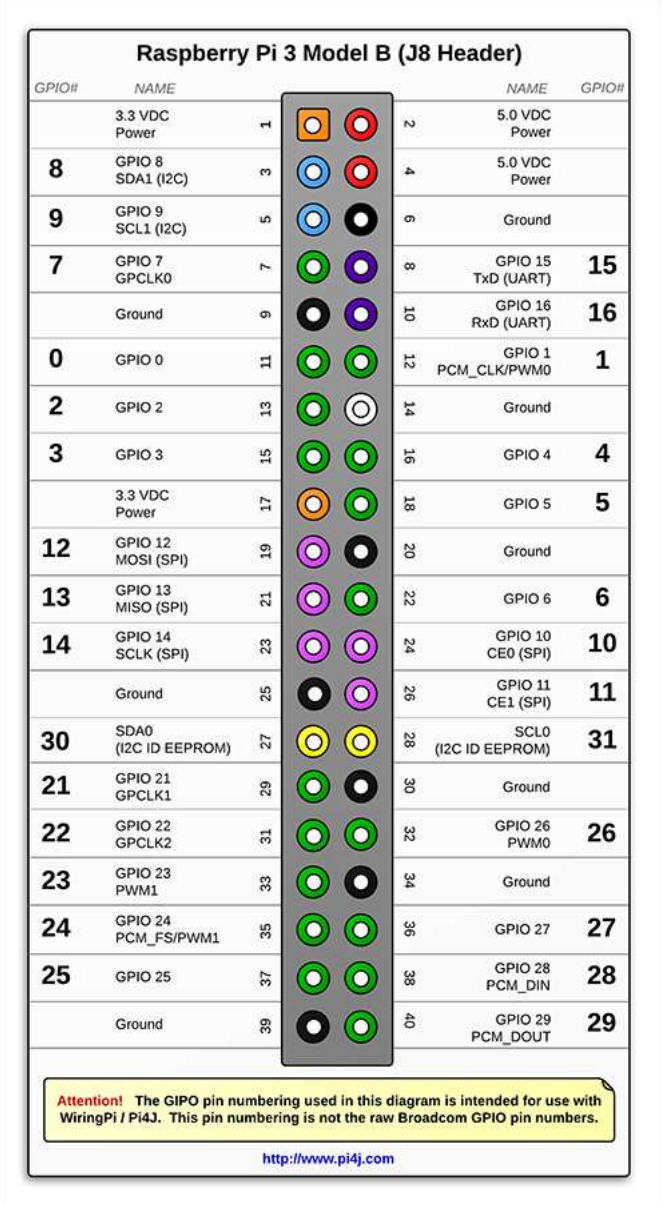


Figura 2.6 Función pines Raspberry Pi 3 Model B

La otra librería fundamental para la aplicación en Java, es una librería gráfica muy amplia y extensa. JFreeChart es una librería de código abierto para Java, la cual permite crear gráficos complejos de forma simple. Esta librería trabaja con GNU Classpath, que es una implementación de software libre de la norma estándar de biblioteca de clases para el lenguaje de programación en Java.

Se pueden utilizar JFreeChart para una diversidad de gráficas grande incluyendo: Gráficos XY, gráficos para series de tiempo, gráficos circulares, diagramas de Gannt, gráficos de barras, histogramas, gráficas específicas, etc. (Ver Ejemplo 1 en Figura 2.10)

Las posibilidades de esta librería en cuanto a gráficos son muchas y permite diseñar la aplicación de la nariz con libertad de elección a la hora de representar los datos de los sensores. Además de la diversidad de gráficas disponibles, es posible colocar marcadores, hacer cambios en la configuración de la gráfica, dibujar automáticamente ejes, escalas y leyendas. (Ver ejemplos 1 y 2 en Figuras 2.11 y 2.12).

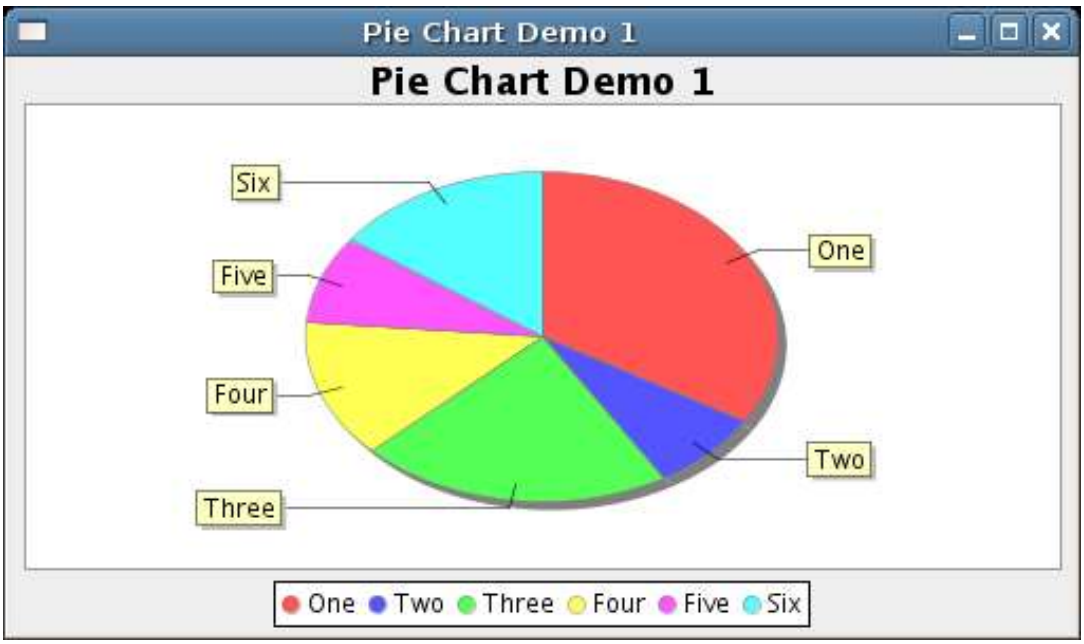


Figura 2.7 Ejemplo 1 JFreeChart

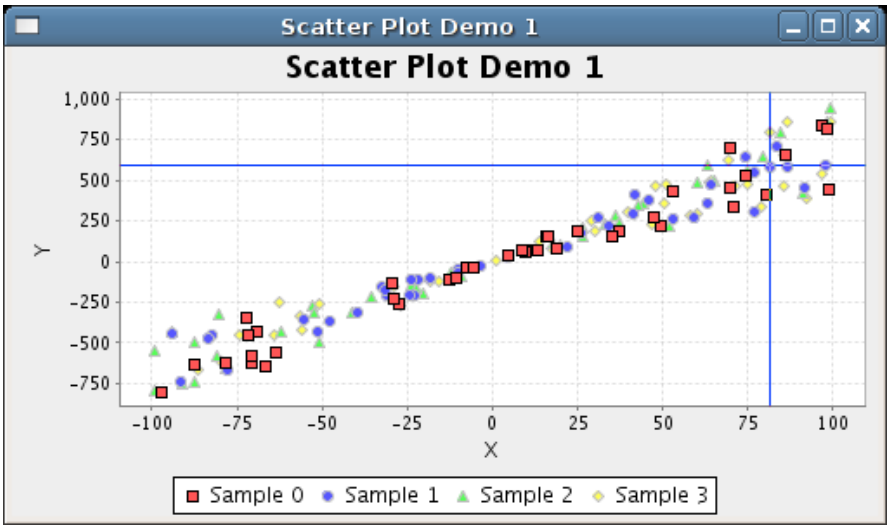


Figura 2.8 Ejemplo 2 JFreeChart

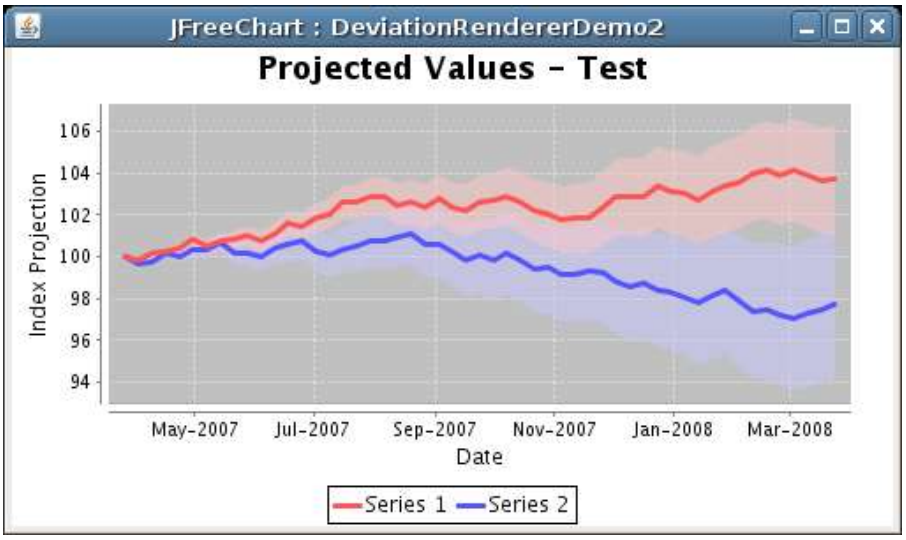


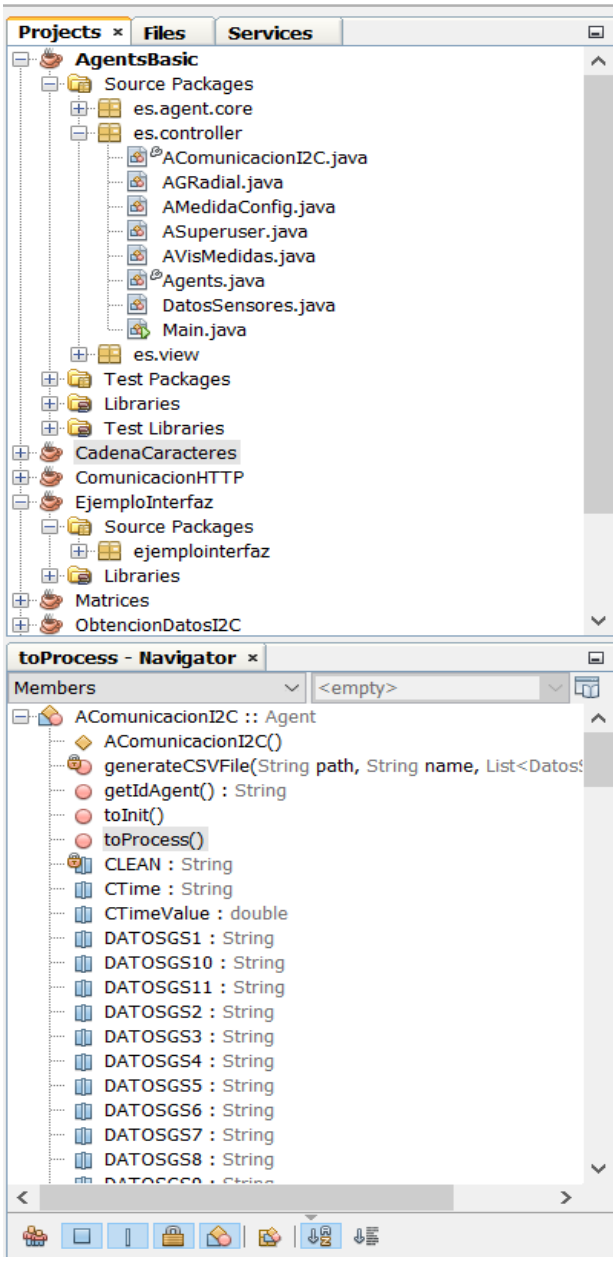
Figura 2.9 Ejemplo 3 JFreeChart

El software de la nariz electrónica tiene una componente fuerte de representación de datos en distinta formas. En primer lugar se usa la librería JFreeChart para hacer una evolución temporal de la respuesta de los sensores al flujo de aire contenedor de volátiles que se hace pasar a través de ellos. El segundo tipo de representación que se hace de los datos es para obtener una huella característica de la calidad del aceite y se usa un diagrama polar.

2.1.3. NetBeans

Netbeans es un entorno de desarrollo integrado libre, principalmente hecho para el lenguaje de programación Java. NetBeans IDE es el producto libre y gratuito sin

restricciones de uso que nos permite crear aplicaciones de escritorio, móviles y web. El IDE de NetBeans ofrece soporte en varios lenguajes, incluyendo por supuesto Java, pero también HTML5, PHP y C++. El entorno ayuda íntegramente en todo el ciclo de desarrollo, desde la creación del proyecto, pasando por depuración y compilación. Es compatible y puede ser instalado en cualquier sistema operativo con Windows, Linux, Mac OS y otros sistemas basados en UNIX.



El IDE proporciona soporte para JDK 8 y las recientes mejoras de Java. Es el primer entorno de programación que proporciona soporte para JDK 8, Java EE 7, y JavaFX 2.

La filosofía de NetBeans es crear código de manera rápida e inteligente. No se trata solo de un editor de texto, el editor de NetBeans introduce líneas, une

Figura 2.10 Organizador de proyectos NetBeans

palabras y corchetes, y resalta el código fuente sintáctica y semánticamente. Le permite refactorizar fácilmente el código, con una gama de herramientas útiles y potentes, mientras que también proporciona plantillas de códigos, sugerencias de codificación y generadores de códigos.

Otra característica que simboliza la facilidad del entorno, es su eficiencia en la distribución y agrupación de proyectos (Ver Figura 2.13). Mantiene una vista limpia de una gran cantidad de aplicaciones, proyectos, carpetas y archivos en la ventana

de vista general. NetBeans ofrece diferentes posibilidades para ver y organizar los archivos que sin duda ayudan al desarrollo de código eficiente y organizado.

El uso de interfaces Java programadas desde el entorno NetBeans es otra ventaja de este editor. Crear y diseñar una GUI para Java SE es una tarea rápida y sencilla porque existe la opción de arrastrar las herramientas que queremos utilizar directas a nuestra pantalla de trabajo. Para las aplicaciones Java SE, NetBeans GUI Builder se ocupa de aplicar automáticamente el espaciado y la alineación correctamente, a la vez que admite la edición in situ.

Es una herramienta tan potente que permitiría hablar mucho más en detalle de ella y extendería demasiado este documento. Por ello remito a cualquier interés que pueda surgir o a usar de guía el documento con referencia en la bibliografía: NetBeans Developing Applications with NetBeans IDE. Es un documento muy extenso donde se puede encontrar toda la información necesaria para exprimir todo el jugo del entorno de programación NetBeans IDE. También fue referencia durante el tiempo de desarrollo de la programación de la nariz electrónica.

NetBeans (Ver Figura 2.14) no es el único entorno de programación con el que programar en Java. Existe por ejemplo Eclipse, también una plataforma de código abierto multiplataforma. Está compuesto por un conjunto de herramientas de programación para desarrollar entornos de desarrollo integrado, IDE, igual que en NetBeans. Aunque existen diferencias entre ambos, no soy demasiado relevantes para el objetivo que se perseguía y cualquiera de los dos IDEs hubiera permitido concluir en los objetivos marcados.

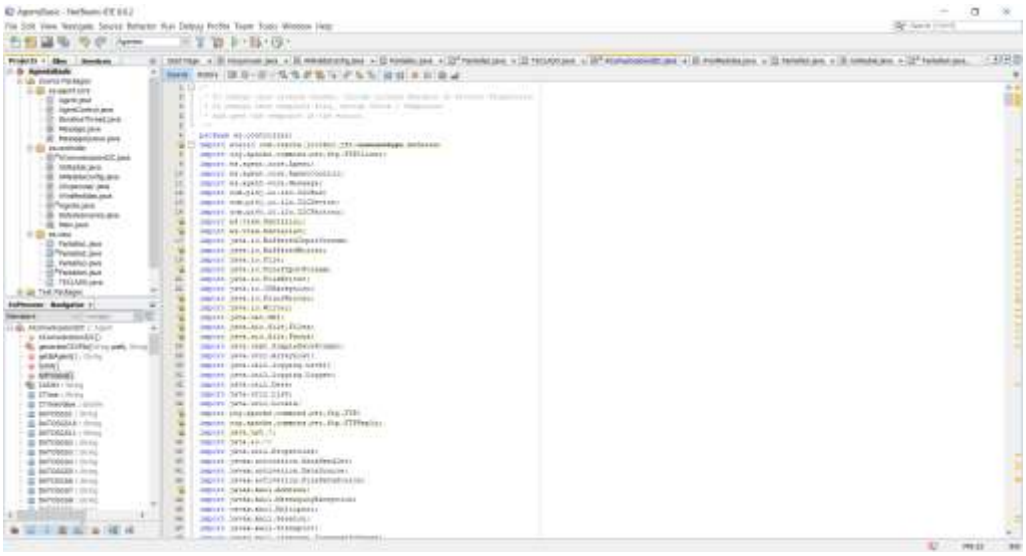


Figura 2.11 Entorno NetBeans IDE

2.1.4. Runtime Remote Platform: Raspberry y NetBeans

Uno de los problemas que hubo durante el proceso de la puesta a punto de las plataformas de desarrollo fue la tarea de conjugar dos de los componentes ya explicados en apartados anteriores. NetBeans es un programa pesado como para que Raspberry pueda hacerse cargo de él y del procesamiento que conlleva programar en Java. Crear código, depurar y compilar no es una no se podía hacer todo en el propio mini PC porque se agotarían los recursos del procesador y se haría mucho más lento todo el desarrollo.

La solución pasaba por crear todo el código en un PC con más capacidad que la Raspberry, y a continuación utilizar una plataforma remota de compilación, que perfectamente podía ser asumido por la Raspberry. De esa forma la programación en Java, además de ser más cómoda utilizar un portátil personal, era mucho más liviana para el microcontrolador de la nariz y así funcionar más rápido y mejor.

La herramienta usada para dicho propósito fue ‘Remote Plarform’ facilitada por NetBeans IDE. Se trata de una solución sencilla que permite programar y construir el proyecto en el ordenador y posteriormente enviar y lanzar el proyecto construido en la Raspberry. Esta opción viene instalada en las versiones de NetBeans 8.0 o superiores, además se necesita que la Raspberry tenga instalado JavaFX, el cual viene por defecto y por último que ambos dispositivos, portátil personal y placa Raspberry estén conectados a internet en la misma red.

En nuestro caso, hemos usado una conexión de ambos dispositivos punto a punto con un cable Ethernet, debido a los problemas que da la red universitaria Eduroam en este tipo de conexiones. Muchos de los puertos necesarios para este tipo de conexiones están cerrados por motivos de seguridad informática.

Una vez que se envía un proyecto desde nuestro ordenador personal a la Raspberry Pi, se crea en ésta un archivo .jar que contiene todo el código de nuestro proyecto. Para lanzar el proyecto solo hay que abrir la consola y llamar al proyecto con los comandos necesarios para ello: Primero, posicionarnos en la carpeta con ‘cd’. En segundo lugar, comprobar la fecha de actualización de proyecto con ‘ls –l’ y por último, lanzar la aplicación Java con ‘java –jar’ más el nombre del proyecto.

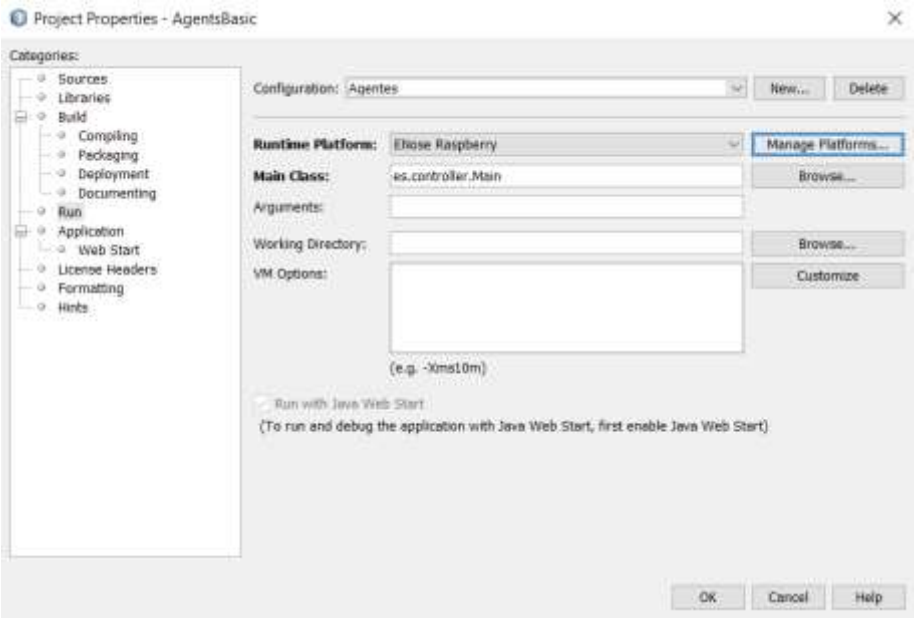


Figura 2.12 Propiedades del proyecto, Runtime Platform

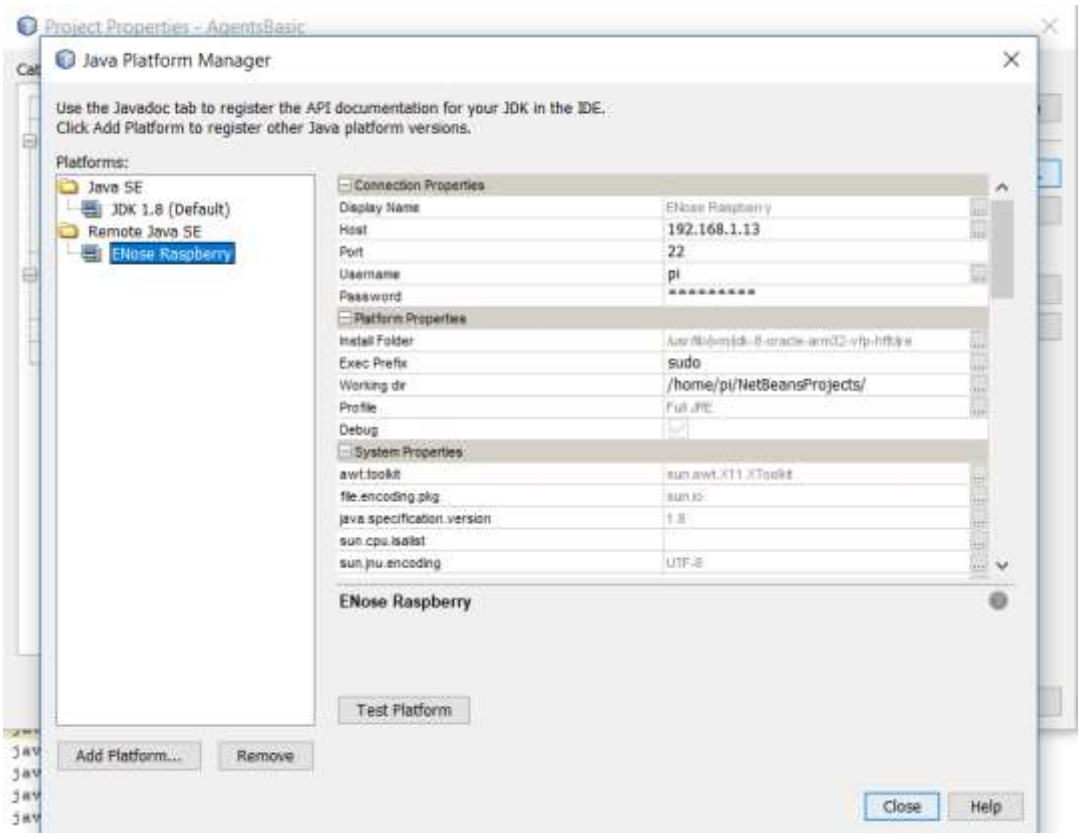


Figura 2.13 Java Platform Manager, ENose Raspberry

En el Anexo 7 se explica paso por paso cómo realizar una instalación completa de las plataformas que hemos utilizado para crear proyectos de la misma arquitectura y características del software de la nariz electrónica.

2.1.5. Conexiones SSH: Putty y Xming

Enlazando con el apartado anterior, en este se trata el uso remoto de la Raspberry desde un ordenador personal. En el apartado dedicado a Raspberry se ha visto que ésta puede ser utilizada con facilidad a través de sus periféricos USB, es decir, con un teclado y un ratón USB. Y con una pantalla HDMI para visualizar la interfaz del sistema operativo Raspbian. Pero realmente para un uso más eficiente es mejor obviar el teclado, el ratón y la pantalla supletoria y utilizar nuestro propio portátil como único periférico de todo el sistema. Desde nuestro portátil abriríamos un portal desde el que controlar a la Raspberry. Como una especie de máquina virtual que usa conexión SSH para funcionar.

SSH o Secure Shell es un protocolo de administración remota que permite controlar y modificar los servidores del usuario a través de conexión a internet. Utiliza técnicas criptográficas para garantizar la seguridad de las comunicaciones hacia y desde el servidor remoto, de forma que éstas estén encriptadas. SSH ofrece un procedimiento para autenticar a un usuario concreto de forma remota, transferir entradas desde el cliente al host y retransmitir la salida de vuelta al cliente.

La conexión SSH utiliza distintas técnicas de cifrado, las cuales son la principal ventaja frente a otros tipos de comunicaciones a través de internet. Se asegura la transferencia de información cifrada entre el host y el cliente. En el caso de estudio el host sería la Raspberry Pi, mientras que el cliente sería el equipo portátil. Hay tres tecnologías de cifrado utilizadas para el SSH:

- Encriptación simétrica (Ver Figura 2.17).

Se utiliza una clave secreta tanto para el cifrado como para el descifrado de un mensaje en servidor y en cliente.

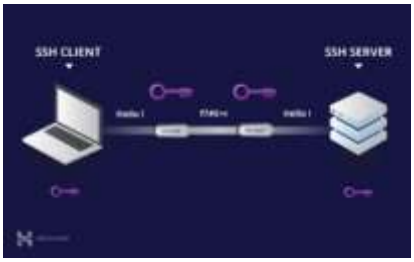


Figura 2.14 SSH: Cifrado simétrico

- Encriptación asimétrica (Ver Figura 2.18).

Utiliza dos claves separadas para el cifrado y el descifrado, conocidas como ‘public key’ y ‘private key’. Las dos claves forman el par de claves público-privado.

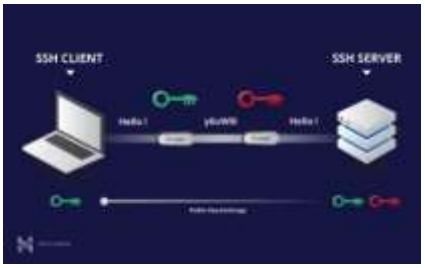


Figura 2.15 SSH: Cifrado asimétrico

- Hashing (Ver Figura 2.19).

Se trata de otra forma de criptografía. El hashing unidireccional nunca está destinado a ser descifrado. Se genera un valor único de una longitud fija para cada entrada que no muestra una tendencia clara que pueda explotarse.

Figura 2.16 SSH: Hashing

Como ya hemos dicho, SSH funciona con el modelo cliente-servidor. Opera en el puerto 22 de forma predeterminada, es decir, el servidor escucha en el puerto 22 las conexiones entrantes. A continuación se inicia la conexión encriptada y la autenticación del cliente si la verificación es correcta. Es el cliente el que debe iniciar la conexión SSH presentando las credenciales de usuario predeterminadas.

En Windows el cliente SSH más utilizado y popular es PuTTY y es precisamente el que hemos utilizado en nuestro desarrollo. PuTTY es un cliente SSH con licencia libre exclusivo para Windows. El nombre PuTTY proviene de las siglas Pu: Port unique TTY: terminal type. Cuya traducción al español sería, Puerto único de tipo terminal.

Lo que permite PuTTY es controlar la Raspberry por comandos desde el dispositivo Windows y de esta manera realizar ciertas acciones sin necesidad de conectar otros periféricos a la Raspberry. Las ventajas de utilizar esta herramienta son muchas:

- Gratuito y de código abierto
- Disponible en Windows pero en desarrollo para Linux
- Aplicación portable
- Interfaz sencilla
- Completo y flexible con muchas opciones de uso

Figura 2.17 PuTTY

Se puede descargar PuTTY fácilmente desde su página oficial. En el Anexo 7 se explicará paso por paso todas las configuraciones.

A pesar de que PuTTY resuelve buena parte de los problemas en cuanto a autonomía del uso de Raspberry desde nuestro equipo, se puede ir más allá y controlar también la Raspberry desde el ordenador personal tal y como si tuviéramos conectados una pantalla, un ratón y un teclado.

Para este propósito se utiliza la herramienta Xming. Xming es un servidor de implementación portátil de ventanas X para sistemas operativos Windows. Es ideal para combinar con PuTTY y así conseguir el control completo de las sesiones cliente-servidor con conexión SSH. De esta manera es muy sencillo obtener un entorno gráfico de Linux, Raspbian en este caso, en Windows y trabajar con él sin necesidad de habilitar otro equipo. Y además, el entorno gráfico al que permite conectarnos Xming es mucho más amigable que la ventana de comandos que habilita PuTTY.

El único requisito necesario para que esta configuración funcione es que los dispositivos estén conectados a la misma red y realizar la conexión SSH con PuTTY y Xming. Aunque para empezar a trabajar parece una configuración compleja, a la hora de desarrollar código Java en el portátil e ir probando y viendo resultados en Raspberry, la eficiencia es muy alta con esta arquitectura.

Figura 2.18 Xming

Por último y antes de acabar con este apartado, mostraremos brevemente el tipo de conexión entre dispositivos que hemos realizado. Aunque lo ideal sería conectar ambos equipos sin cables, a través de una red WiFi, hacerlo de esta manera resulta muy difícil en redes con puertos cerrados. Por eso la conexión con cable Ethernet desde el PC hasta el mini PC es la solución utilizada (Ver Figura 2. 22). Es posible conectar directamente la Raspberry Pi al PC utilizando un cable de red CAT-5. La velocidad de conexión es de hasta 100 Mbps y no es necesario tener un cable punto a punto o cruzado, ya que por lo general las tarjetas de red de los nuevos portátiles y la Raspberry Pi son Auto-Sense (Auto-MDI) y automáticamente selecciona el tipo de conexión.

Figura 2.19 Conexión punto a punto para comunicación SSH entre dispositivos

2.2. Adquisición de datos

Tradicionalmente un sistema de adquisición de datos se entiende como el proceso de medir con un software controlador una variable o fenómeno del medio físico. Un sistema de adquisición de datos consiste en un grupo de sensores que miden en su entorno físico, el hardware de adaptación en la toma de datos de los sensores y un bus comunicante al PC con su software de aplicación (Ver Figura 2. 23).

Un sensor es un transductor que convierte un fenómeno físico en una señal eléctrica que se puede medir. Dependiendo del tipo de sensor, su salida puede ser tensión, corriente o resistencia. Algunos sensores necesitan adaptación especial de su señal para ser procesados, otros necesitan adaptación de la alimentación para su adquisición, en definitiva, detrás de la adquisición de datos hay un trabajo importante de la parte hardware. El hardware actúa de interfaz entre la controladora y el entorno físico exterior. Las señales del entorno físico son naturalmente analógicas y en consecuencia hay que digitalizarlas para su interpretación.

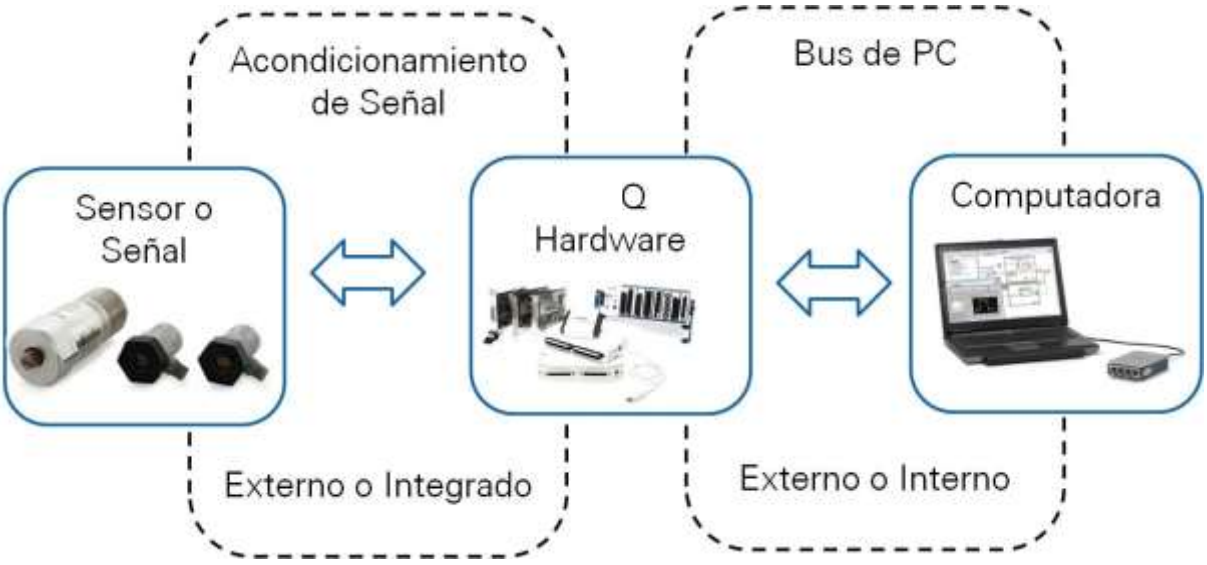


Figura 2.20 Esquema de un sistema completo de adquisición de datos

Esto se hace principalmente en tres componentes del sistema:

- Circuito de acondicionamiento de las señales

Las señales de los sensores pueden ser ruidosas y turbias. El circuito de adaptación o de acondicionamiento manipula la señal para que pueda llegar a ser entrada del convertidor ADC. El circuito de acondicionamiento puede amplificar la señal, atenuarla, hacerla pasar por condensadores de filtrado, etc.

- Convertidor analógico-digital

Antes de que la señal llegue al software controlador, la señal analógica debe ser convertida en digital. Un convertidor es un componente electrónico que proporciona una representación digital de la señal analógica de entrada.

- Bus de datos

Las señales digitales que salen del convertidor, ya están preparadas para lanzarse al bus de datos y ser transferidas al PC para su presentación. Digamos que el bus de datos es el vaso comunicante entre los componentes hardware que tratan la señal y el PC.

La última parte de un sistema de adquisición de datos corresponde precisamente con lo que en la parte software de la nariz electrónica se ha desarrollado en la adquisición de datos. Hasta ahora todo lo explicado correspondía a la parte hardware

de la nariz electrónica, y el software se utiliza para procesar, visualizar y almacenar los datos de medida. En el caso de la nariz electrónica, el PC dedicado a esta función es la Raspberry Pi. La aplicación con funcionalidad personalizada que interactúa entre el PC y el usuario es la programada en Java, preparada para adquirir, analizar y presentar datos de las medidas. También automatiza procesos propios relacionados con la medida de los sensores y realiza algoritmos de procesamiento de señales. El bus de datos escogido para interactuar entre el medio físico de la cámara de sensores y el software es la comunicación I2C, de la que trataremos en el siguiente punto.

2.2.1. Comunicación I2C

I2C son siglas en inglés: Inter-Integrated Circuit. Es un protocolo de comunicación serial desarrollado por Phillips Semiconductors en el año 1982. Su uso principalmente recae en la comunicación de diferentes partes de un circuito, controlador y periféricos. Inicialmente la idea era comunicar distintas placas internamente en un televisor, pero la tecnología se fue desarrollando pasando por varios estados hasta llegar a lo que es hoy, una integración de otros protocolos como el SPI y UART. I2C se basa en un modelo maestro-esclavo, es capaz de tener en el bus a varios maestros y varios esclavos a la vez, o como en nuestro caso, un maestro controlando varios esclavos.

El protocolo I2C tiene este diseño eléctrico de la Figura 2. 24. Utiliza dos cables de comunicación que son la señal de reloj (SCL, Serial Clock) y la línea de datos (SDA, Serial Data). Las dos líneas necesitan resistencias conocidas como ‘pull-up’ para su conexión con el cable de alimentación Vdd. El bus de datos I2C trabaja con lógica positiva, un nivel alto en el bus se traduce en un 1 lógico, y un nivel bajo en un 0 lógico.

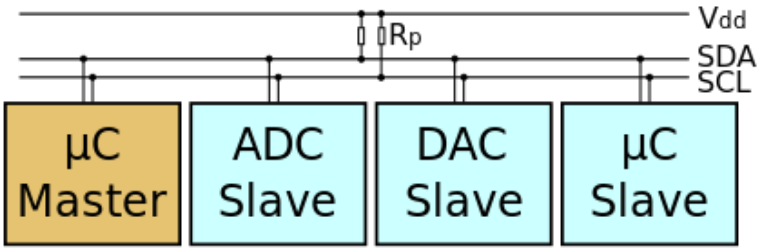


Figura 2.21 Esquema maestro y tres nodos esclavos

Se puede observar que el protocolo de comunicación envía los datos a través de una sola línea de comunicación, y la información es enviada bit a bit de manera

coordinada. Se puede decir que es un protocolo de comunicación serial. Además trabaja de forma síncrona. El envío de datos por la vía SDA está sincronizado por una señal de reloj que comparten maestros y esclavos en la línea SCL.

La siguiente tabla muestra una ficha técnica resumida de las características del protocolo de comunicación I2C.

Número de vías o cables	2
Velocidad máxima	Modo estándar (Sm) = 100kbps
	Modo rápido (Fm) = 400kbps
	Modo High Speed (Fm+) = 3.4Mbps
	Modo Ultra Fast (Hs-mode) = 5Mbps
Síncrono o Asíncrono	Síncrono
Paralelo o Serial	Serial
Número máximo de Maestros	Ilimitado
Número máximo de Esclavos	1008

Tabla 2.4 Ficha técnica protocolo I2C

Para trabajar con varios nodos esclavos es importante conocer la dirección de estos. La dirección por defecto que establece el protocolo es la de un primer byte enviado por el maestro, donde los 7 primeros bits representan la dirección y el octavo es el que comunica al esclavo si debe recibir datos del maestro, nivel bajo, o si debe enviar datos al maestro, nivel alto. Esto genera un espacio de direccionamiento de 7 bits, lo cual permite hasta 112 nodos en un bus. Hay 16 direcciones reservadas a fines especiales. Posteriormente se ha introducido una ampliación de direccionamiento de 10 bits para alcanzar una capacidad de 1136 nodos en un bus.

Con el protocolo de comunicación I2C un microcontrolador como Raspberry puede controlar toda una red de periféricos asociados. Simplifica mucho el proceso de obtención de datos ya que el bus solo tiene dos líneas, y el número de pines que tendría un circuito integrado se reduce considerablemente, reduciendo también el tiempo y el coste. Se considera un sistema de bajo coste aunque sea más lento que otros protocolos de comunicación. Uno de los mayores problemas del I2C y que puede

llegar a ser un problema para la nariz electrónica es la susceptibilidad que presenta a interferencias, como puede ser la compatibilidad electromagnética.

Resumimos las características generales del protocolo de comunicación I2C en sus ventajas y desventajas:

Ventajas

- Esquema simple con solo dos cables de comunicación.
- Soporta sistemas multimaestros y múltiples esclavos.
- Hardware de acondicionamiento sencillo.
- Protocolo extensamente conocido y utilizado.

Desventajas

- Comunicación más lenta que otro tipo de comunicaciones.
- Limitado tamaño de paquetes de información.
- Posibles interferencias con ruido electromagnético en el bus.

La Raspberry Pi dispone de dos periféricos para implementar I2C. En primer lugar el BSC (Broadcom Serial Controller) que implementa el modo maestro. Y en segundo lugar el BSI (Broadcom Serial Interface) que implementa el modo esclavo. La nariz electrónica utiliza la Raspberry como maestro, por tanto solo se utiliza el BSC.

El BSC implementa tres maestros diferentes e independientes unos de otros. El BSC0, el BSC1 y el BSC2. El único maestro al que se accede desde la aplicación de la nariz es al BSC1, mediante los pines 3 y 5 del conector J8 de la Raspberry Pi. Los maestros BSC0 y BSC2 están destinados a la identificación de placas de expansión y el puerto HDMI respectivamente (Ver Figura 2.25)

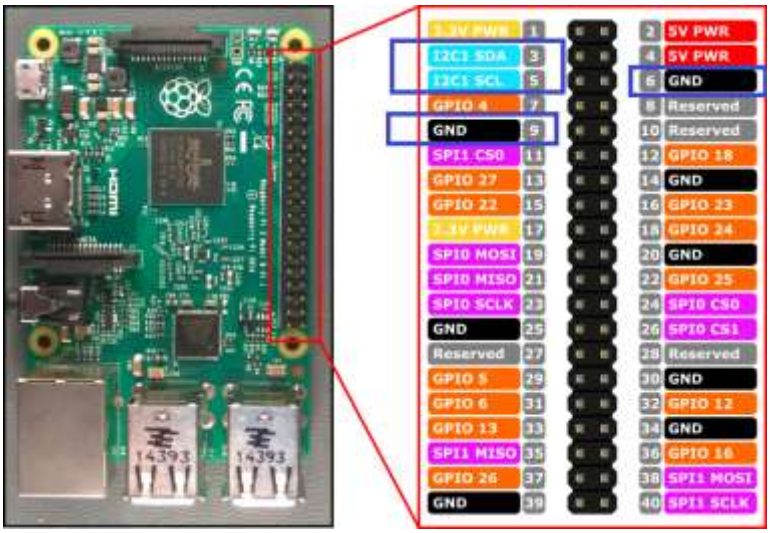


Figura 2.22 Conector J8 con pines 3 y 5, SDA y SCL

Los dispositivos, llamados esclavos, conectados al bus de comunicaciones I2C tienen una dirección física con una longitud de 7 bits. Se podría utilizar también el direccionamiento mencionado anteriormente de 10 bits, pero en este caso se direccionan los dispositivos con 7 bits.

Existen comandos para la detección de interfaces I2C disponibles y para visualizar los dispositivos conectados al bus. Esto es muy interesante a la hora de hacer pruebas con los dispositivos conectados, si uno de ellos falla o, una vez conectado, no se encuentra en el bus.

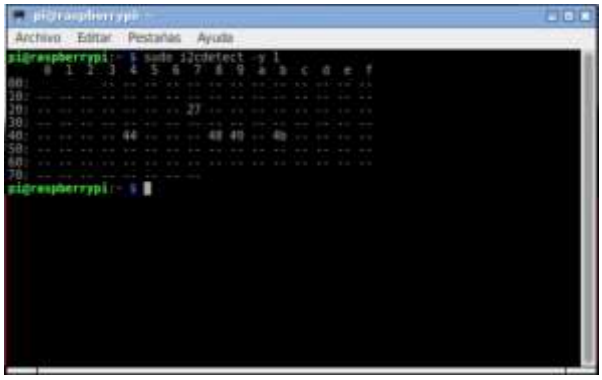


Figura 2.23 Dispositivos conectados al bus de datos en la Raspberry

Arriba, en la Figura 2. 26, se pueden ver cinco direcciones físicas que tienen que ver con los dispositivos de los que trataremos en los siguientes puntos. La Raspberry controla las cinco direcciones con el protocolo I2C y envía o recibe información según la función del dispositivo.

En la dirección 27 del bus se encuentra el multiplexor MCP23008 conectado a la placa de relés. En la dirección 44 del bus está el sensor de temperatura y húmeda SHT31. Y por último, los tres convertidores ADS1115 que recogen los datos de todos los sensores de la placa de sensores están direccionados en las direcciones 48, 49 y 4B.

Antes de pasar a explicar en detalle cada dispositivo, mostraremos cómo se inicia en Java cada dirección I2C con el fin de ejemplificar lo que acabamos de comentar.

```
@Override
public synchronized void toProcess() throws IterativeThreadException, InterruptedException {
    try {
        // Create I2CBus CONVERTIDOR 1
        I2CBus busC1 = I2CFactory.getInstance(I2CBus.BUS_1);
        I2CDevice deviceC1 = busC1.getDevice(0x48);

        // Create I2CBus CONVERTIDOR 2
        I2CBus busC2 = I2CFactory.getInstance(I2CBus.BUS_1);
        I2CDevice deviceC2= busC2.getDevice(0x49);


        // Create I2CBus CONVERTIDOR 3
        I2CBus busC3 = I2CFactory.getInstance(I2CBus.BUS_1);
        I2CDevice deviceC3= busC3.getDevice(0x4B);

        // Create I2CBus
        I2CBus busSHT = I2CFactory.getInstance(I2CBus.BUS_1);
        // Get I2C device, SHT31 I2C address is 0x44(68)
        I2CDevice deviceSHT = busSHT.getDevice(0x44);

           Create I2C bus
        I2CBus busMCP = I2CFactory.getInstance(I2CBus.BUS_1);
        // Get I2C device, MCP23008 I2C address is 0x20(32)
        I2CDevice deviceMCP = busMCP.getDevice(0x27);
```

Figura 2.24 Creación en Java de los dispositivos I2C

2.2.1.1. Convertidor ADS1115

El ADS1115 es un convertidor analógico-digital de precisión, baja potencia, compatible con I2C y de 16 bit de resolución de Texas Instruments. Está encapsulado en SMD y tiene cuatro canales de adquisición, por tres convertidores que tiene la nariz electrónica, tenemos capacidad para adquirir doce datos de diferentes sensores. El convertidor ADS1115 permite programar la ganancia amplificador e incluye un comparador digital, estas características y el amplio rango de tensiones de alimentación hacen a este convertidor ideal para la medida de sensores.

Es capaz de hacer conversión con tasas de hasta 860 muestras por segundo. La amplificación de ganancia (PGA) ofrece rangos de entrada desde ± 256 mV hasta ± 6.144 V, permitiendo medidas de gran y pequeña señal. Esta característica permite diseñar cada canal del convertidor respecto a las necesidades del sensor a medir. Aunque hay que decir que, independientemente del PGA elegido, la máxima tensión que podemos medir será siempre la tensión de alimentación. Es decir, aunque el PGA lo configuremos a 6.144V, no podremos medir tensiones superiores a nuestra tensión de alimentación.

Sin profundizar más en el funcionamiento de muestreo del convertidor, nos centraremos solamente en la programación del mismo, realizada en Java, para la obtención de datos.

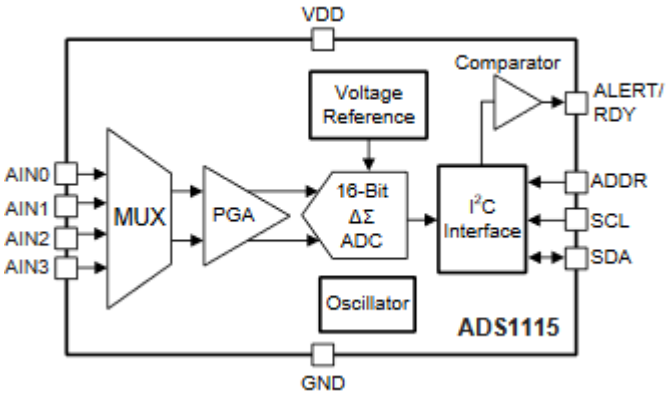


Figura 2.25 Diagrama de bloques del convertidor ADS1115

El ADS1115 se comunica a través de la interfaz I2C (Figura 2.28). Está diseñado de tal manera que dos dispositivos que quieran comunicar a través del bus al mismo tiempo no entren en conflicto. Cuando hay una comunicación entre dos dispositivos, uno actúa de maestro y otro de esclavo como hemos visto antes. Este convertidor solo puede actuar como esclavo y concretamente su función en la nariz es solo de enviar datos, en ningún caso de recibir.

El convertidor tiene cuatro registros que son accesibles por medio de la interfaz I2C usando el registro de punteros de dirección (Ver Figura 2. 29). El registro de conversión contiene el resultado de la última conversión. El registro de configuración se usa para cambiar los modos de operación del convertidor y consulta el estado del dispositivo. Los otros dos registros, ‘Lo_thresh’ y ‘Hi_thresh’ establecen los valores límites usados por la función del comparador.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	P[1:0]	
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -n = value after reset

Figura 2.26 Registro puntero de direcciones

Bit	Field	Type	Reset	Description
7:2	Reserved	W	0h	Always write 0h
1:0	P[1:0]	W	0h	**Register address pointer** 00 : Conversion register 01 : Config register 10 : Lo_thresh register 11 : Hi_thresh register

Tabla 2.5 Descripción del registro puntero de direcciones

El registro necesario editar para la programación de los modos de operación del convertidor es el registro de configuración (Ver Figura 2.30). Los demás registros del convertidor no se han cambiado desde la programación.

El registro de configuración es un registro de 16 bit en el que se puede cambiar:

- Los modos de operación.
- Selección de la configuración de la entrada.
- La tasa de muestreo.
- El rango de fondo de escala.
- Y los modos de comparación.

15	14	13	12	11	10	9	8
OS	MUX[2:0]			PGA[2:0]			MODE
R/W-1h	R/W-0h			R/W-2h			R/W-1h
7	6	5	4	3	2	1	0
DR[2:0]			COMP_MODE	COMP_POL	COMP_LAT	COMP_QUE[1:0]	
R/W-4h			R/W-0h	R/W-0h	R/W-0h	R/W-3h	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Figura 2.27 Registro de configuración del convertidor

Para elegir la configuración que se desea, basta con escribir en el registro el número en hexadecimal correspondiente a la combinación binaria que cumple con la configuración que queremos. Por defecto tiene valores dados en todos los bits, por ejemplo, ±2.048 V en la ganancia de amplificación. Las agrupaciones de bit que ha habido que programar han sido los bit 11, 10 y 9 y los bits 14, 13 y 12. En la siguiente tabla se puede ver la combinación de 1s y 0s necesaria para establecer cada parámetro.

Bit	Field	Type	Reset	Description
15	OS	R/W	1h	**Operational status or single-shot conversion start** This bit determines the operational status of the device. OS can only be written when in power-down state and has no effect when a conversion is ongoing. When writing: 0 : No effect 1 : Start a single conversion (when in power-down state) When reading: 0 : Device is currently performing a conversion 1 : Device is not currently performing a conversion
14:12	MUX[2:0]	R/W	0h	**Input multiplexer configuration (ADS1115 only)** These bits configure the input multiplexer. These bits serve no function on the ADS1113 and ADS1114. 000 : AIN_P = AIN0 and AIN_N = AIN1 (default) 001 : AIN_P = AIN0 and AIN_N = AIN3 010 : AIN_P = AIN1 and AIN_N = AIN3 011 : AIN_P = AIN2 and AIN_N = AIN3 100 : AIN_P = AIN0 and AIN_N = GND 101 : AIN_P = AIN1 and AIN_N = GND 110 : AIN_P = AIN2 and AIN_N = GND 111 : AIN_P = AIN3 and AIN_N = GND
11:9	PGA[2:0]	R/W	2h	**Programmable gain amplifier configuration** These bits set the FSR of the programmable gain amplifier. These bits serve no function on the ADS1113. 000 : FSR = ±6.144 V(1) 001 : FSR = ±4.096 V(1) 010 : FSR = ±2.048 V (default) 011 : FSR = ±1.024 V 100 : FSR = ±0.512 V 101 : FSR = ±0.256 V 110 : FSR = ±0.256 V 111 : FSR = ±0.256 V
8	MODE	R/W	1h	**Device operating mode** This bit controls the operating mode. 0 : Continuous-conversion mode 1 : Single-shot mode or power-down state (default)
7:5	DR[2:0]	R/W	4h	**Data rate** These bits control the data rate setting. 000 : 8 SPS 001 : 16 SPS 010 : 32 SPS 011 : 64 SPS 100 : 128 SPS (default) 101 : 250 SPS 110 : 475 SPS 111 : 860 SPS

4	COMP_MODE	R/W	0h	**Comparator mode (ADS1114 and ADS1115 only)** This bit configures the comparator operating mode. This bit serves no function on the ADS1113. 0 : Traditional comparator (default) 1 : Window comparator
3	COMP_POL	R/W	0h	**Comparator polarity (ADS1114 and ADS1115 only)** This bit controls the polarity of the ALERT/RDY pin. This bit serves no function on the ADS1113. 0 : Active low (default) 1 : Active high
2	COMP_LAT	R/W	0h	**Latching comparator (ADS1114 and ADS1115 only)** This bit controls whether the ALERT/RDY pin latches after being asserted or clears after conversions are within the margin of the upper and lower threshold values. This bit serves no function on the ADS1113. 0 : Nonlatching comparator . The ALERT/RDY pin does not latch when asserted (default). 1 : Latching comparator. The asserted ALERT/RDY pin remains latched until conversion data are read by the master or an appropriate SMBus alert response is sent by the master. The device responds with its address, and it is the lowest address currently asserting the ALERT/RDY bus line.
1:0	COMP_QUE[1:0]	R/W	3h	**Comparator queue and disable (ADS1114 and ADS1115 only)** These bits perform two functions. When set to 11, the comparator is disabled and the ALERT/RDY pin is set to a high-impedance state. When set to any other value, the ALERT/RDY pin and the comparator function are enabled, and the set value determines the number of successive conversions exceeding the upper or lower threshold required before asserting the ALERT/RDY pin. These bits serve no function on the ADS1113. 00 : Assert after one conversion 01 : Assert after two conversions 10 : Assert after four conversions 11 : Disable comparator and set ALERT/RDY pin to high-impedance (default)

Tabla 2.6 Descripción del registro de configuración

Como ejemplo de programación de un canal del convertidor, mostramos una parte del código de obtención de datos de un sensor en la Figura 2.31:

```
//case sensor 1: A-SP35, 0X48 CH1
case AComunicacionI2C.DATOSS1:
    if (flag51) {

   // Get I2C device, ADS1115 I2C address is 0x48(72)

      byte[] config1 = {(byte)0xC0, (byte)0x83};
      // Select configuration register
      // AINP = AIN0 and AINN = GND, +/- 2.048V, Continuous conversion mode, 128 SPS
      deviceC1.write(0x01, config1, 0, 2);
      Thread.sleep(25);

      // Read 2 bytes of data
      // raw_adc msb, raw_adc lsb
      byte[] data1 = new byte[2];
      deviceC1.read(0x00, data1, 0, 2);

      // Convert the data
      raw_adc1 = ((data1[0] & 0xFF) * 256) + (data1[1] & 0xFF);
      if (raw_adc1 > 32767)
      {
          raw_adc1 -= 65535;
      }
```

Figura 2.28 Ejemplo de obtención de datos del convertidor desde Java

La ventaja de trabajar con Open Hardware es que existen muchas librerías ya preparadas para multitud de dispositivos. En este caso Adafruit, que también fabrica y comercializa este convertidor, aporta mucha información sobre diagramas eléctricos, código fuente, esquemas PCB, etc.

2.2.1.2. Multiplexor MCP23008

El dispositivo MCP23X08 proporciona una expansión de E/S paralela de 8 bits de propósito general para el bus I2C o las aplicaciones SPI. Los dos dispositivos difieren en el número de pines de dirección de hardware y la interfaz en serie:

- MCP23008 - interfaz I2C; tres pines de dirección
- MCP23S08 - interfaz SPI; dos pines de dirección

El MCP23008 de I2C es el que hemos usado (Ver Figura 2.32). Consiste en múltiples registros de configuración de 8 bits para la selección de entrada, salida y polaridad. El maestro del sistema puede habilitar las E/S como entradas o salidas escribiendo los bits de configuración de E/S. Los datos para cada entrada o salida se mantienen en el registro de Entrada o Salida correspondiente. La polaridad del registro del puerto de entrada se puede invertir con el registro de inversión de polaridad. Todos los registros pueden ser leídos por el maestro del sistema.

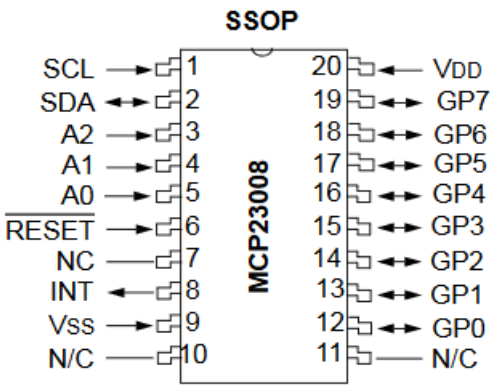


Figura 2.29 Diagrama MCP23008

El multiplexor en el sistema de la nariz electrónica se utiliza para dar salidas a los relés mecánicos y activarlos o desactivarlos según convenga.

Los relés están asociados a las electroválvulas neumáticas, la bomba que insufla el aire y la alimentación de los sensores. Los pines SCL y SDA se conectan al bus de datos I2C. El direccionamiento del multiplexor se realiza igual que el convertidor, con 7 bits, con el bit de lectura/escritura en el último bit del byte de control. La dirección del MCP contiene cuatro bits fijos y tres bits de dirección de hardware definidos por el usuario (pines A2, A1 y A0 de la Figura 2.33).

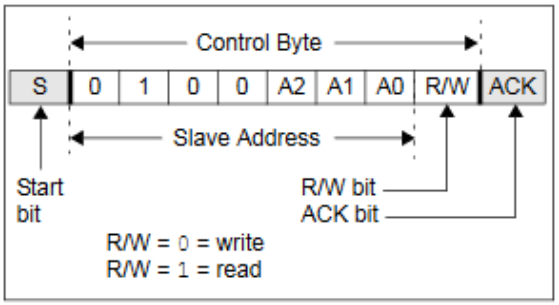


Figura 2.30 Byte de control MCP23008

A modo de ejemplo, en la Figura siguiente reproducimos una muestra de la activación de un relé desde Java.

```
// Create I2C bus
I2CBus busMCP;
try {
    busMCP = I2CFactory.getInstance(I2CBus.BUS_1);
    // Get I2C device, MCP23008 I2C address is 0x20(32)
    I2CDevice deviceMCP = busMCP.getDevice(0x27);
    deviceMCP.write(0x00, (byte)0x00);

    reles = (byte)(0x00);
    reles = (byte)(reles | (1<<0));  // ACTIVAR
    deviceMCP.write(0x09, (byte)reles);
```

Figura 2.31 Activación de un relé por software

2.2.1.3. Sensor temperatura y humedad SHT31

SHT3x-DIS pertenece a la próxima generación de sensores de temperatura y humedad de Sensirion. Se basa en un nuevo chip sensor CMOSensor, el cual ha aumentado la inteligencia, la fiabilidad y las especificaciones de precisión, destacablemente mejoradas en comparación con su predecesor. Su funcionalidad incluye un procesamiento de señal mejorado, dos direcciones I2C distintas y seleccionables por el usuario y velocidades de comunicación de hasta 1 MHz.

La placa que se ve en la Figura 2. 35 es una placa de adaptación fabricada por Adafruit para el sensor de temperatura y humedad SHT31 de Sensirion.

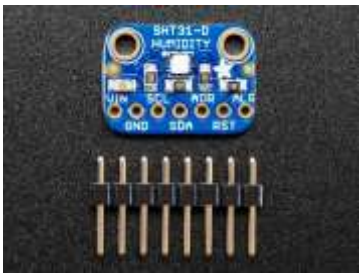


Figura 2.32 SHT31

Este sensor se usa para medir la temperatura y la humedad a la que se encuentra la cámara de sensores. Estos parámetros son muy útiles a la hora de analizar los datos y su influencia en el comportamiento de las curvas de los sensores puede ser notoria como demuestran algunos estudios. (Beltrán Ortega y otros, 2015)

Figura 2.33 Configuración de pines sensor SHT31

La programación en Java de este sensor para obtener su respuesta es sencilla y de manera muy similar a como se obtienen los datos del convertidor. Además Adafruit también cuenta con librerías propias de 'open source' y con las cuales se hace mucho más fácil la utilización de hardware a la soluciones software. El sensor es capaz de dar como salida tres valores, la temperatura en grados Celsius y Fahrenheit con ±0.3 grados de precisión, y la humedad relativa con un ±2% de precisión.

A continuación se muestra un ejemplo de la obtención de las variables desde Java.

```
//case sensor 12: SHT31
case AComunicacionI2C.DATOSSHTTemp:
    if (flagS12) {

        // Send high repeatability measurement command
        // Command msb, command lsb
        byte[] configSHT1 = new byte[2];
        configSHT1[0] = (byte)0x2C;
        configSHT1[1] = (byte)0x06;
        deviceSHT.write(configSHT1, 0, 2);
        Thread.sleep(25);

        // Read 6 bytes of data
        // temp msb, temp lsb, temp CRC, humidity msb, humidity lsb, humidity CRC
        byte[] dataSHT = new byte[6];
        deviceSHT.read(dataSHT, 0, 6);

        //Convert the data
        cTemp = ((((dataSHT[0] & 0xFF) * 256) + (dataSHT[1] & 0xFF)) * 175.0) / 65535.0  - 45.0;
        //double fTemp = ((((dataSHT[0] & 0xFF) * 256) + (dataSHT[1] & 0xFF)) * 315.0) / 65535.0 - 49.0;
        //humidity = ((((dataSHT[3] & 0xFF) * 256) + (dataSHT[4] & 0xFF)) * 100.0) / 65535.0;
```

Figura 2.34 Obtención de datos del sensor SHT31 con Java

2.2.2. Adaptación de los datos

En todo sistema de adquisición de datos hay una adaptación de los mismos para su correcta representación. Los sensores de gas de los que se recogen los datos de la nariz electrónica tienen valores muy dispares, son valores absolutos que provienen de la medida de la resistencia interna según cómo varíe en función de los volátiles que capten en cada instante de tiempo.

Al ser los datos tan dispares, el software está programado de manera que permite obtener datos absolutos y datos referenciados. Datos absolutos corresponden a la medida sin adaptación del sensor, y los datos referenciados corresponden a los datos del sensor que han sido adaptados para mejorar su representación. La adaptación de los datos en ningún caso se traduce en una pérdida de información, sino en una transformación de la información con el fin de hacerlos comprensibles, darles sentido y facilitar el análisis posterior de los datos.

A continuación se muestra una respuesta típica de los sensores con los datos adaptados y sin adaptar (Ver Figuras 2.38 y 2.39). Se puede observar que otra de las mayores diferencias es la variabilidad en la excitación de los sensores: los datos adaptados muestran una mayor representatividad de la excitación en la respuesta. Sin embargo, en los datos absolutos no se puede apreciar la variabilidad de los datos, o, al menos, los resultados son confusos.

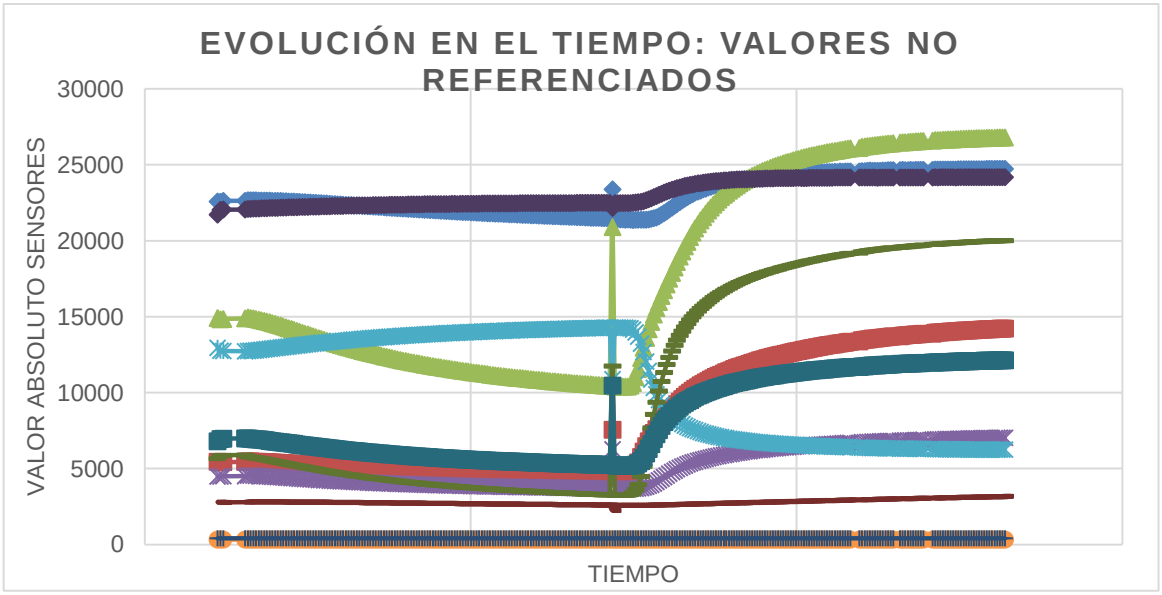


Figura 2.35 Respuesta típica datos absolutos

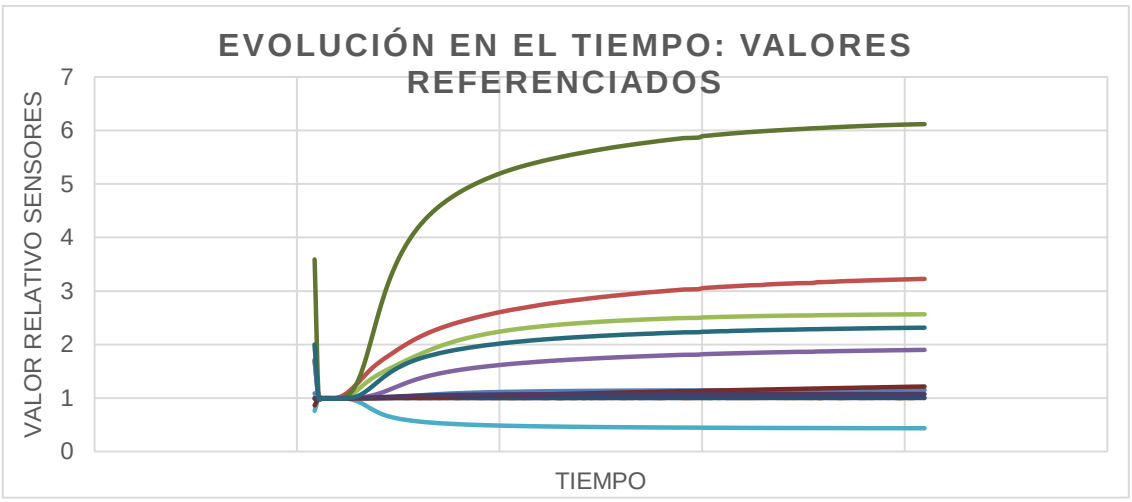


Figura 2.36 Respuesta típica datos referenciados

En el siguiente apartado se explicará todo el funcionamiento del software programado en Java, explicando en detalle cada pantalla y el flujograma interno del control de la nariz electrónica.

2.3. Monitorización y control

Este punto del trabajo es el más importante en la metodología llevada a cabo. Su contenido es el que explica las funciones de monitorización y control del software de la nariz electrónica. Es el resultado del diseño que se ha hecho desde el principio y la carga de trabajo más grande durante los meses de desarrollo.

Ilustración 3 Flujo general del diseño

La estrategia de la programación ha sido estructurada secuencialmente, cumpliendo hitos y marcas dentro del eje temporal del desarrollo. Antes de comenzar con la explicación de cada parte del código en Java, mostraremos un árbol de la arquitectura final del código. Como ya aclaramos en apartados anteriores, la programación en Java está basada en agentes, conforme aclara la siguiente ilustración:

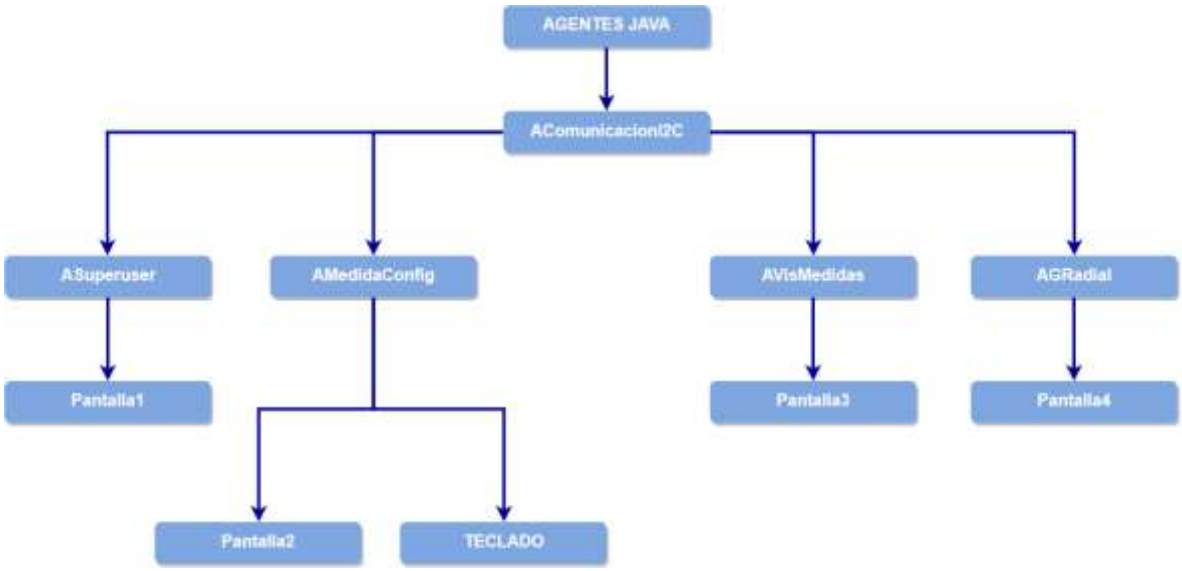


Ilustración 4 Diseño final de agentes y sus clases asociadas

El diseño final está compuesto por cinco agentes que controlan todo el flujo del código. Los agentes se distinguen por una A mayúscula antes de nombre propio del agente. Cada agente tiene su función bien definida y controla la parte que le corresponde. Fijándonos en el árbol, el agente que está en la parte central llamado AComunicacionesI2C es el que tiene comunicación directa con todos los demás agentes y por tanto controla el flujo del programa por completo, se denomina agente primario. Este agente no tiene asociada ninguna clase extra, pero se encarga de realizar múltiples tareas según los demás agentes se las vayan solicitando.

Los otros cuatro agentes sí tienen asociadas clases, que son las pantallas de la interfaz gráfica de usuario. Esta interfaz está compuesta por cuatro pantallas en las que el usuario puede ir desplazándose al utilizar la nariz electrónica (Ver Ilustración 5).

En el árbol también se puede ver que uno de los agentes secundarios, llamado AMedidaConfig, tiene dos clases asociadas. Esto es porque realmente hay dos

pantallas detrás de este agente, Pantalla2 y TECLADO, que es un teclado táctil para que el usuario pueda introducir un correo electrónico.

Como ya hemos dicho anteriormente, cada módulo de programación se ha desarrollado secuencialmente. Hasta que no se acababa la tarea con un módulo de programación no se comenzaba el siguiente. Normalmente cada módulo integraba un agente y su clase asociada, manteniendo así también el orden en la creación de clases.

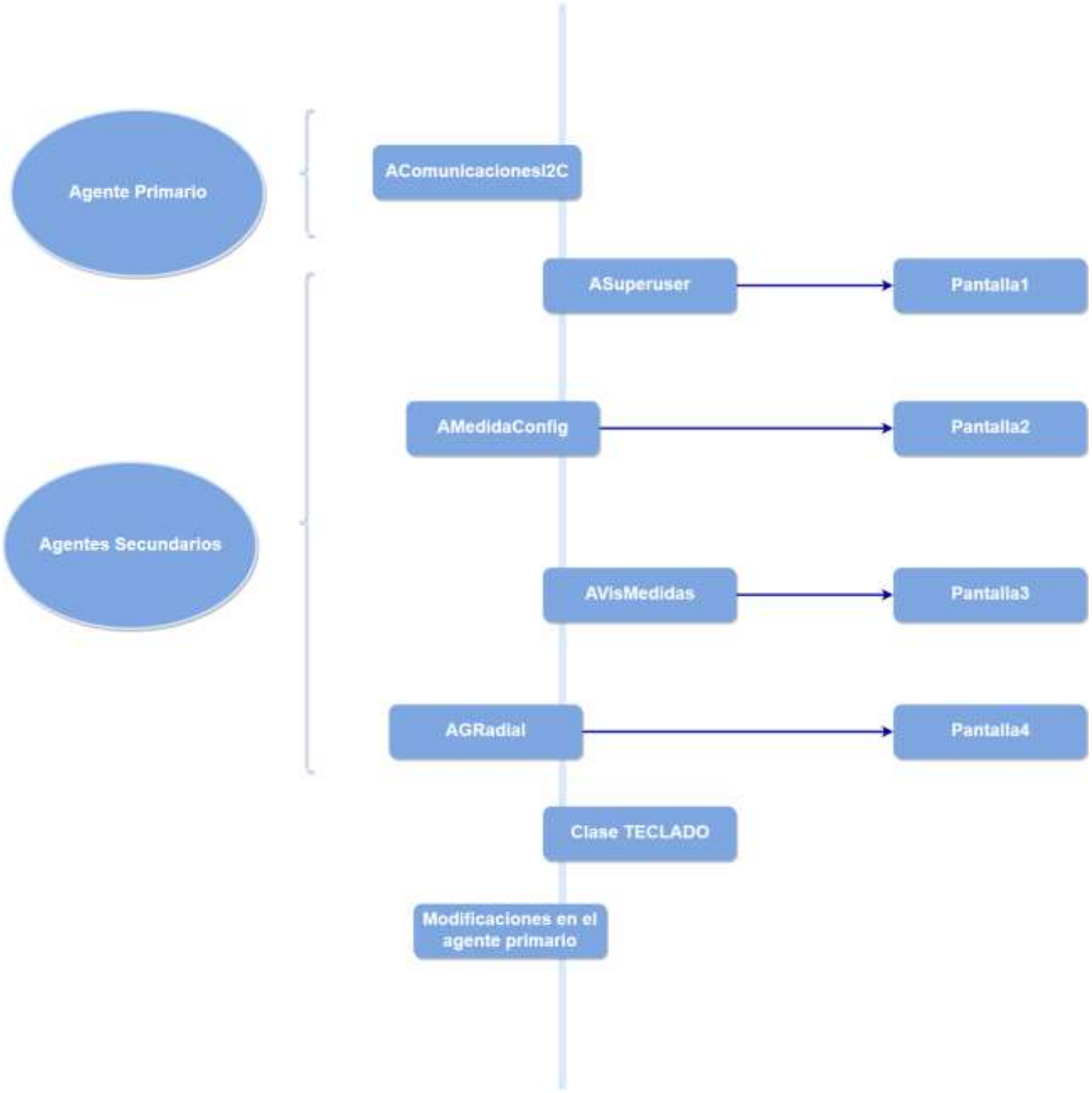


Ilustración 5 Eje temporal del diseño y desarrollo software

Toda la interfaz de la nariz electrónica se muestra en un monitor táctil integrado en la Raspberry, ya que se trata de que la nariz sea una máquina con todas las funcionalidades industriales.

2.3.1. Monitor Táctil

La pantalla física táctil, es decir, la pantalla donde se monitorizan los datos en tiempo real y donde se controla a nivel usuario la nariz electrónica, se considera a nivel de programación un elemento que afecta de forma indirecta a ella. La razón fundamental es el tamaño y la resolución de la pantalla. No es lo mismo diseñar la interfaz sobre una pantalla de 3'' que sobre una de 7''. La pantalla de mayor tamaño permite más flexibilidad y sobretodo deja más espacio a los botones y elementos de control para el usuario.

En una almazara es difícil imaginar que una pantalla pequeña sea realmente útil al operario que está vigilando el proceso de obtención de aceite. Ni las gráficas ni las funciones táctiles de la pantalla serían tarea fácil. Sin embargo es cierto que la elección de la pantalla no solo depende del software, sino que también tiene justificaciones hardware y mecánica. Por este motivo, el prototipo inicial de la nariz electrónica se ha desarrollado con una pantalla más pequeña, concretamente de 4'' y una resolución de 320x480 pixel.

Figura 2.37 Pantalla táctil 4''

Es una pantalla táctil diseñada especialmente para Raspberry Pi. Esta pantalla tiene drivers específicos para el sistema operativo Raspbian, que son los que inicialmente se han instalado en la Raspberry. Esta pantalla da buenos resultados y podría ser totalmente funcional, pero tiene el inconveniente del tamaño. En una pantalla con esa resolución no es fácil manejar la nariz electrónica de manera táctil. Además, esta pantalla, del fabricante Waveshare, necesita una placa de adaptación de sus pines con nuestro sistema y su tiempo de refresco no es demasiado alto.

El siguiente paso fue proponer adquirir para nuestro prototipo final una pantalla de mayor tamaño. Después de buscar pantallas de 7 y hasta de 10'' nos decantamos por una pantalla táctil de 7'' y resolución de 800x480 pixel. Esta pantalla viene con una placa integrada de adaptación que se encarga de la conversión de potencia y señales.

Figura 2.38 Pantalla táctil de 7''

Por tanto en las propiedades del JFrame de la interfaz de Java, inicialmente el tamaño estaba fijado a la resolución de la pantalla pequeña 320x480 pixel, y en el prototipo final está a 800x480 pixel.

2.3.2. Pantalla Superusuario

La pantalla de superusuario es la primera que se explicará en detalle porque fue la primera que se diseñó. Su función inicialmente se pensó como una pantalla accesible

únicamente por contraseña para un superusuario, con el fin de supervisar el estado de la nariz electrónica y sus sensores.

La función final sigue siendo una supervisión de los sensores, del valor que están dando en el momento de la medida. La pantalla está configurada por 13 cuadros de texto en el que se muestra el valor, tipo 'float', de cada sensor (Ver Figura 2.42).

Al lado de cada cuadro de texto, hay una etiqueta con el nombre técnico de cada sensor. Se debatió también el hecho de poner un nombre más comercial a cada sensor para que así pudieran ser más reconocibles por cualquier persona. Sin embargo, el nombre técnico de cada sensor aporta la información completa necesaria para buscar la hoja de características del modelo y conocer sus propiedades. En futuras versiones de la nariz electrónica los nombres de cada sensor podrían pasar a llamarse de una forma más convencional.

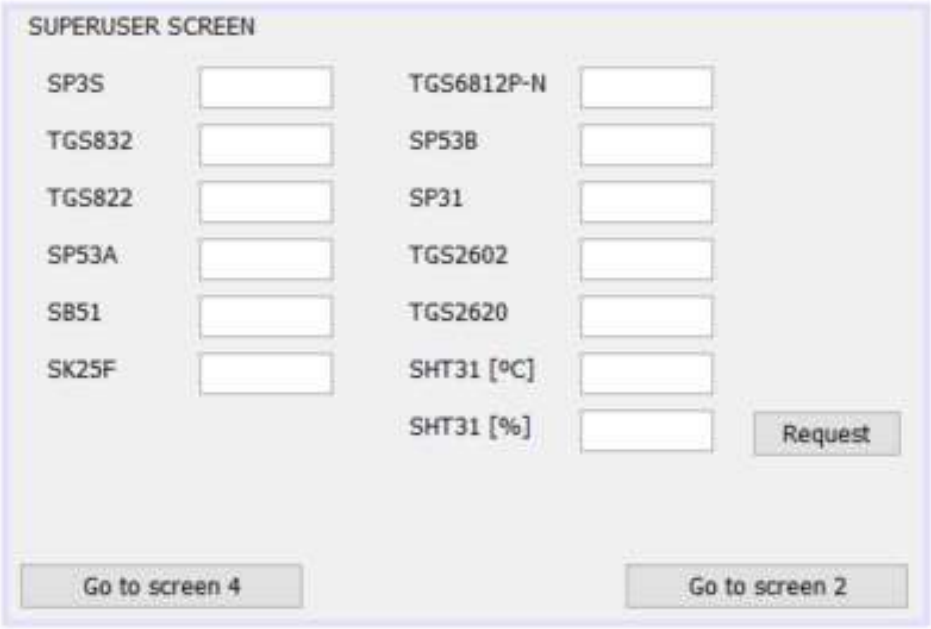


Figura 2.39 Pantalla Superusuario

En esta imagen se puede ver una captura de la interfaz programada en Java. Además de los cuadros de texto y las etiquetas mencionadas anteriormente, se pueden ver tres botones en la interfaz. El botón 'Request' pide los datos de los sensores a otro agente y se colocan cada uno en su lugar. Los botones de la parte inferior de la interfaz sirven para que el usuario se desplace de una pantalla a otra.

La programación de esta pantalla tiene dos partes, el agente y la clase que constituye la interfaz de la imagen superior. Procederemos a explicar con flujogramas cómo funcionan.

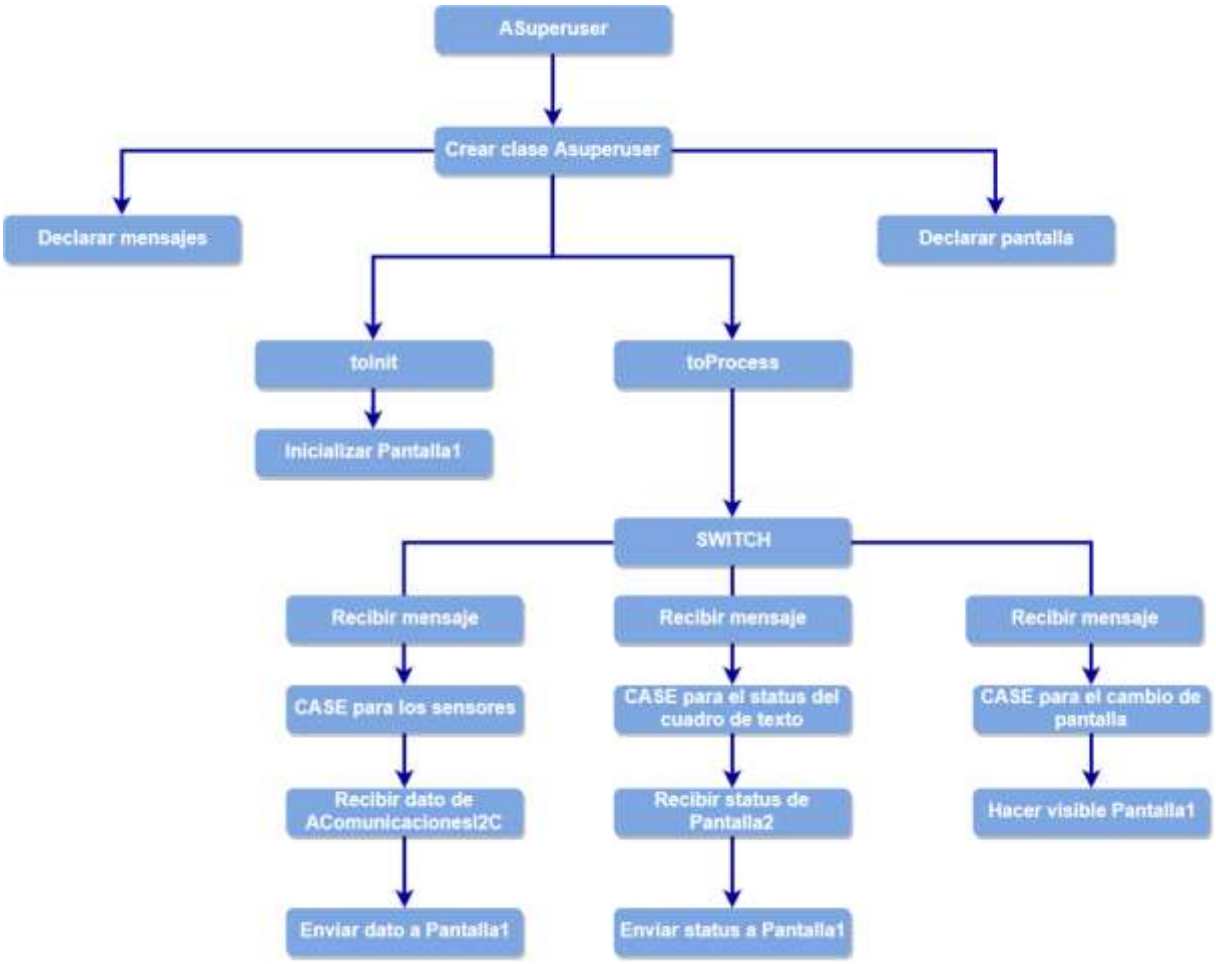


Ilustración 6 Esquema ASuperuser

Lo primero que hay que hacer al crear la clase del agente es declarar los mensajes que vayan a utilizarse desde otros agentes para comunicarse con ASuperuser. Estos tienen una estructura concreta:

```
public class ASuperuser extends Agent{

    public final static String MENSAJE = "mensaje";
    public final static String MENSAJE1 = "mensaje1";
    public final static String MENSAJE3 = "mensaje3";
    public final static String MENSAJE4 = "mensaje4";

    public final static String DATOSSHTTemp = "datosshttemp";
    public final static String DATOSSHTHum = "datosshthum";
    public final static String DATOSS1 = "datoss1";
    public final static String DATOSS2 = "datoss2";
    public final static String DATOSS3 = "datoss3";
    public final static String DATOSS4 = "datoss4";
    public final static String DATOSS5 = "datoss5";
    public final static String DATOSS6 = "datoss6";
    public final static String DATOSS7 = "datoss7";
    public final static String DATOSS8 = "datoss8";
    public final static String DATOSS9 = "datoss9";
    public final static String DATOSS10 = "datoss10";
    public final static String DATOSS11 = "datoss11";
```

Figura 2.40 Ejemplo de declaración de mensajes del agente ASuperuser

A continuación, en el 'toInit' solo se inicializa la Pantalla1. El 'toProcess' es siempre la parte más laboriosa, en ella se estructura el control mediante un 'switch' con todos los 'case' que hagan falta. Cada 'case' tiene la condición del agente de procedencia del mensaje y el mensaje recibido en cuestión. Cuando ASuperuser recibe un mensaje, el flujo entra dentro del 'case' que corresponda. Dentro de cada 'case' se ejecuta el código que queramos programar. Pondremos el ejemplo del código con el cambio entre pantallas de la interfaz en la siguiente Figura:

```
public synchronized void toProcess() throws IterativeThreadException, InterruptedException {
    Message m=getFirstPriorityMessage();
    switch(m.data.property){
        case AMedidaConfig.MENSAJE:
            System.out.println("Mensaje recibido en APantalla1");
            miPantalla.setVisible(true);
            break;

        case AGRadial.MENSAJE1:
            System.out.println("Mensaje recibido en APantalla1");
            miPantalla.setVisible(true);
            break;
```

Figura 2.41 Ejemplo de control de mensajes ASuperuser

Igual que en el ejemplo mostrado, hay 'case' para los datos que se reciben de cada sensor y para hacer editable o no el campo de texto de Pantalla1.

```
case AComunicacionI2C.DATOSS1:
    System.out.println("Dato de sensor 1 recibido");
    miPantalla.setTextS1((double)m.data.value);
    break;
```

Figura 2.42 Envío de datos de sensores a Pantalla1

En la Figura 2.45 vemos que, al entrar en la condición del ‘case’ de un sensor, el código que se ejecuta es el envío del dato del sensor a través de la función ‘setTextS1’, y se corresponde con un método programado en la clase Pantalla1. El dato enviado se ha recibido de AComunicacionI2C y se encuentra en ‘((double)m.data.value)’.

Se procede igualmente en el estado editable del campo de texto.

```
case ASuperuser.CHANGE_STATUS_S1:
    miPantalla.setEnableS1((boolean)m.data.value);
    break;
```

Figura 2.43 Envío del estado del campo de texto a Pantalla1

La segunda parte de la programación del bloque ASuperuser/Pantalla1 es la clase Pantalla1.

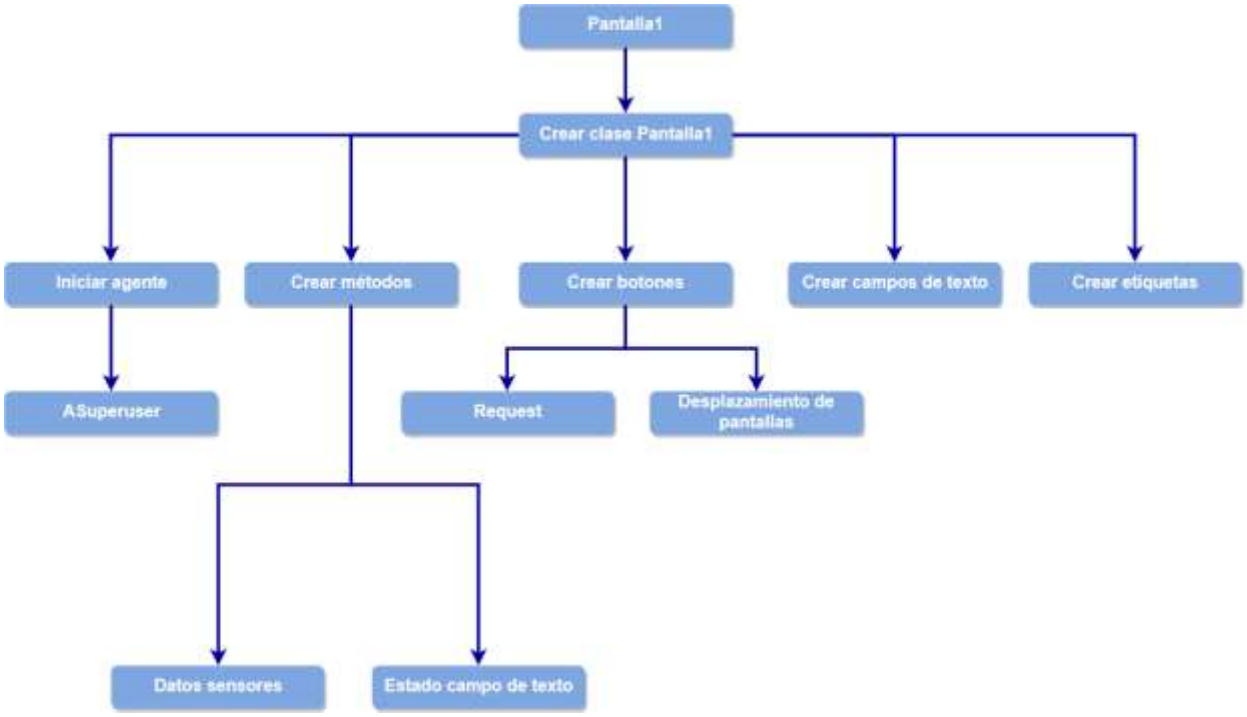


Ilustración 7 Esquema Pantalla1

A las clases asociadas a los agentes hay que decirles a qué clase pertenecen al iniciar la clase. En el caso de Pantalla1 pertenece a ASuperuser y se hace de la siguiente manera.

```
public class Pantalla1 extends javax.swing.JFrame {

    Agent ASuperuser;

    /**
     * Creates new form Pantalla1
     */
    public Pantalla1(Agent agent) {
        initComponents();
        this.ASuperuser = agent;
    }
```

Figura 2.44 Asignación de pertenencia de una clase a un agente

Como ya se ha comentado antes, los datos de cada sensor se colocaban en un campo de texto de la interfaz y se hace mediante una función simple de Java.

```
public void setTextSHTTemp(double datoT){
    jTextField12.setText(String.format("%.2f", datoT,""C"));
}
```

Figura 2.45 Método que introduce el dato del sensor en el campo de texto

'jTextField12' hace referencia a un campo de texto concreto y la función 'setText' añade el valor de datoT tipo doublé como un 'String' con formato de dos decimales.

Para actualizar el estado del campo de texto se hace también a través de un método, pero en este caso el tipo de dato es tipo 'boolean'.

```
public void setEnableS1(boolean state){
    jTextField1.setEnabled(state);
}
```

Figura 2.46 Método que cambia el estado del campo de texto, de editable a no editable

Editar el campo de texto según editable o no editable tiene sentido porque en el siguiente punto veremos que otra pantalla de la interfaz permite seleccionar qué sensores queremos medir, y, por tanto, qué sensores se mostrarán tanto en las representaciones como en estos campos de texto.

Los botones en Java se implementan con el elemento 'jButton'. Tienen una función sencilla:

- **Request**

Solicita los datos de cada sensor al agente primario AComunicacionI2C. Para que un botón haga una acción al pulsarlo, se puede implementar el método automáticamente en el que se ejecuta lo que hay dentro tras pulsar el botón.

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        // TODO add your handling code here:
        AgentControl.sendMessage(ASuperuser, Agents.getAgentComunicacionI2C(), new Message.PropertyValue(AComunicacionI2C.DATOSS1));
```

Figura 2.47 Acción del botón Request y solicitud de datos al agente primario

La solicitud de datos es realmente el envío de un mensaje concreto a otro agente desde el agente en el que se encuentre la acción. En este caso se envía el mensaje DATOSS1 desde ASuperuser a AComunicacionI2C con la estructura de la figura superior. Cuando el mensaje se recibe en AComunicacionI2C, el 'case' correspondiente ejecutará el código que tenga programado.

- **Desplazamiento de pantallas**

Al pulsar el botón de ir de una pantalla a otra, se envía en cada caso un mensaje al agente que controle la pantalla a la que se desplaza el botón. De esta manera se recibe el mensaje en dicho agente y éste hace visible la pantalla como vimos en la Figura 2.44.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        // TODO add your handling code here:
        AgentControl.sendMessage(ASuperuser, Agents.getAgentPantalla2(), new Message.PropertyValue(AMedidaConfig.MENSAJE));

    } catch (Exception e) {
        System.out.println("Excepcion en pantalla1: "+ e);
    }
}
```

Figura 2.48 Acción de los botones de desplazamiento de pantallas

Por último se crean los campos de texto y las etiquetas con los elementos 'jTextField' y 'jLabel' respectivamente.

2.3.3. Pantalla de Configuración de medidas

La pantalla de Configuración de la medida es importante porque es la pantalla inicial de la aplicación de la nariz electrónica (Ver Figura 2. 52). Además es la que ofrece mayor interacción con el usuario. En ella se pueden seleccionar los sensores que se

quieren medir, establecer los tiempos de limpieza y de medida, elegir entre la representación de datos referenciados o absoluta, e incluso elegir si el usuario quiere enviar los datos de la medida por email.

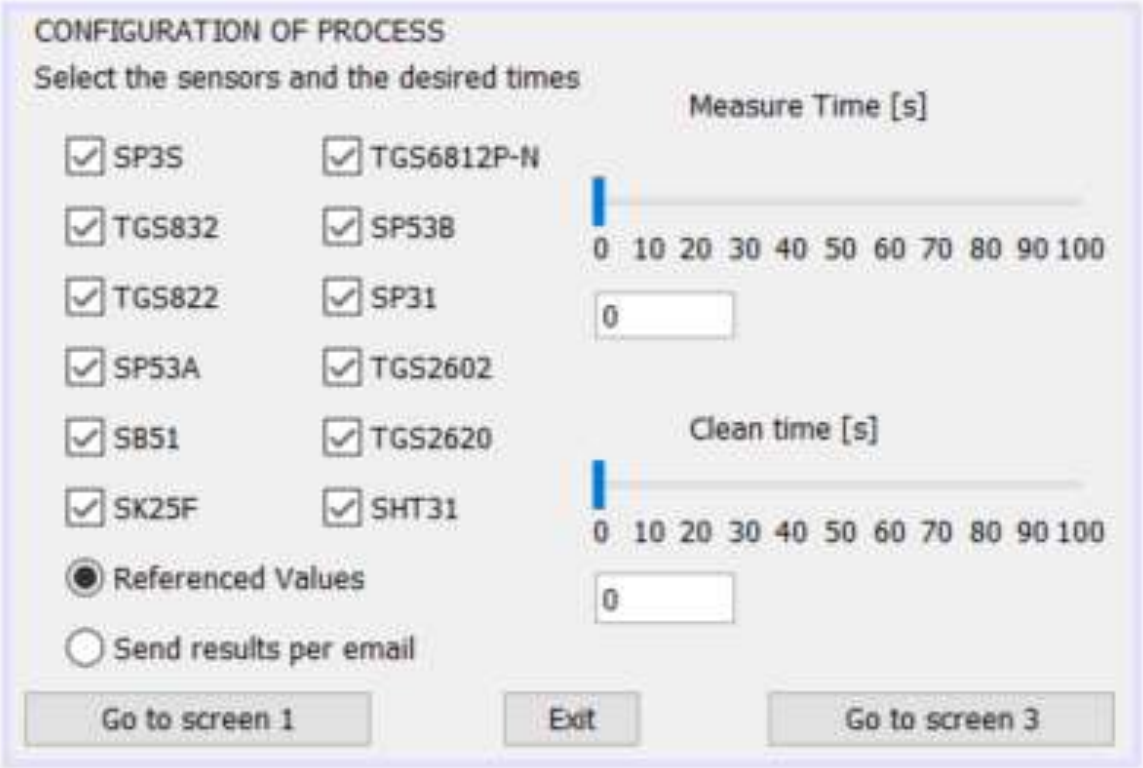


Figura 2.49 Pantalla Configuración de medidas

En la imagen anterior vemos una captura de la interfaz de la pantalla de configuración de la medida. En ella se aprecia un diseño pensado para que el usuario configure la medida de la nariz electrónica. Antes de seguir explicando, merece la pena destacar que el funcionamiento de la nariz consiste en hacer pasar el flujo de aire con contenidos volátiles del aceite a través de los sensores. Para que los sensores den medidas fiables, es importante que, además del flujo de aire con volátiles, por los sensores se haga pasar un flujo de aire limpio durante un determinado tiempo siempre antes de empezar a medir de nuevo. Es por esta razón por la cual la interfaz permite configurar dos tiempos, el de medida y el de limpieza.

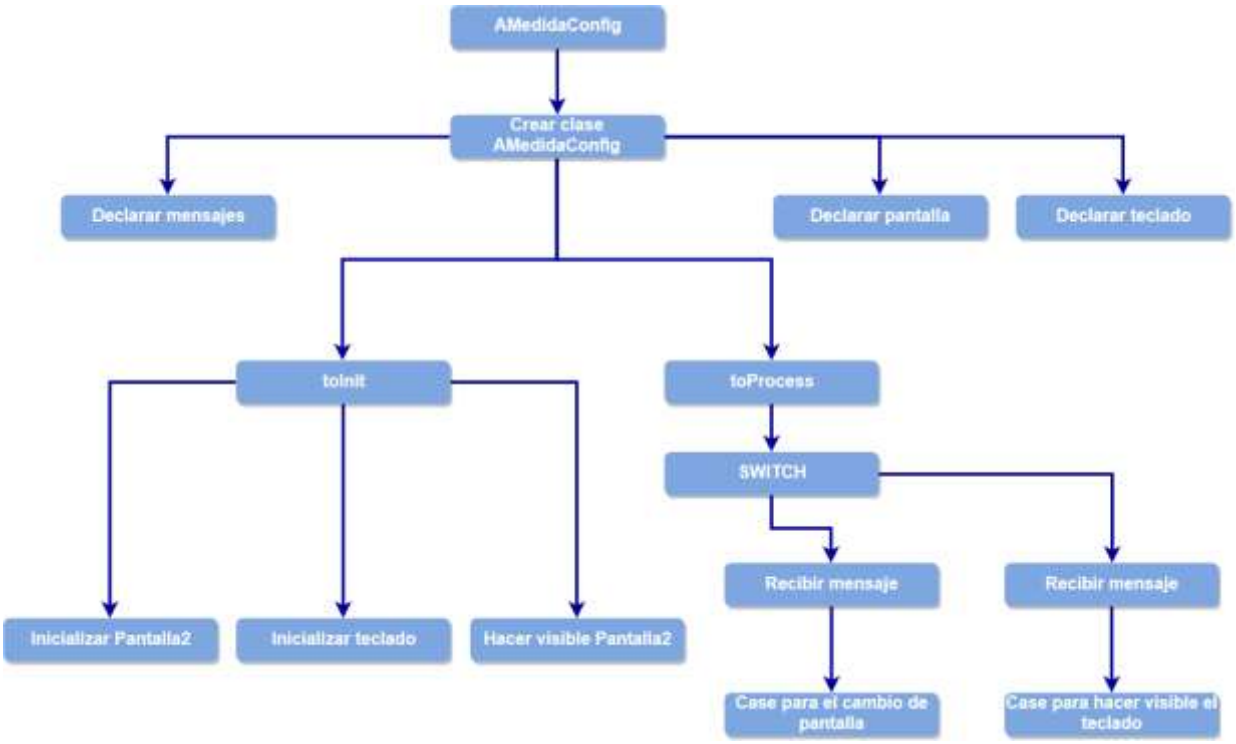


Ilustración 8 Esquema AMedidaConfig

Al igual que en el agente anterior, se deben declarar los mensajes necesarios. El agente AMedidaConfig (Ilustración 8) es el único que tiene dos clases asociadas, una pantalla y un teclado. El teclado realmente es otra pantalla que aparece al elegir la opción de enviar los resultados por email. En el 'toInit', por tanto, hay que inicializar las dos.

```
public synchronized void toInit() throws IterativeThreadException {
    System.out.println("Agente pantalla de configuración de medida iniciado.");
    miPantalla = new Pantalla2(this);
    miTeclado = new TECLADO (this);
    miPantalla.setVisible(true);
}
```

Figura 2.50 Declaración de la pantalla y el teclado en el agente AMedidaConfig

El 'toProcess' de este agente es mucho más sencillo que el anterior. Bien es cierto que hay más carga de código en sus clases asociadas y por eso el agente queda más liberado.

El 'switch' de control solo debe contemplar tres 'case'. Los dos de desplazamiento de pantalla para hacer visible Pantalla2 cuando llegue un mensaje de Pantalla1 o Pantalla3. Y el 'case' de hacer visible la pantalla TECLADO cuando se elija la opción de enviar resultados por correo.

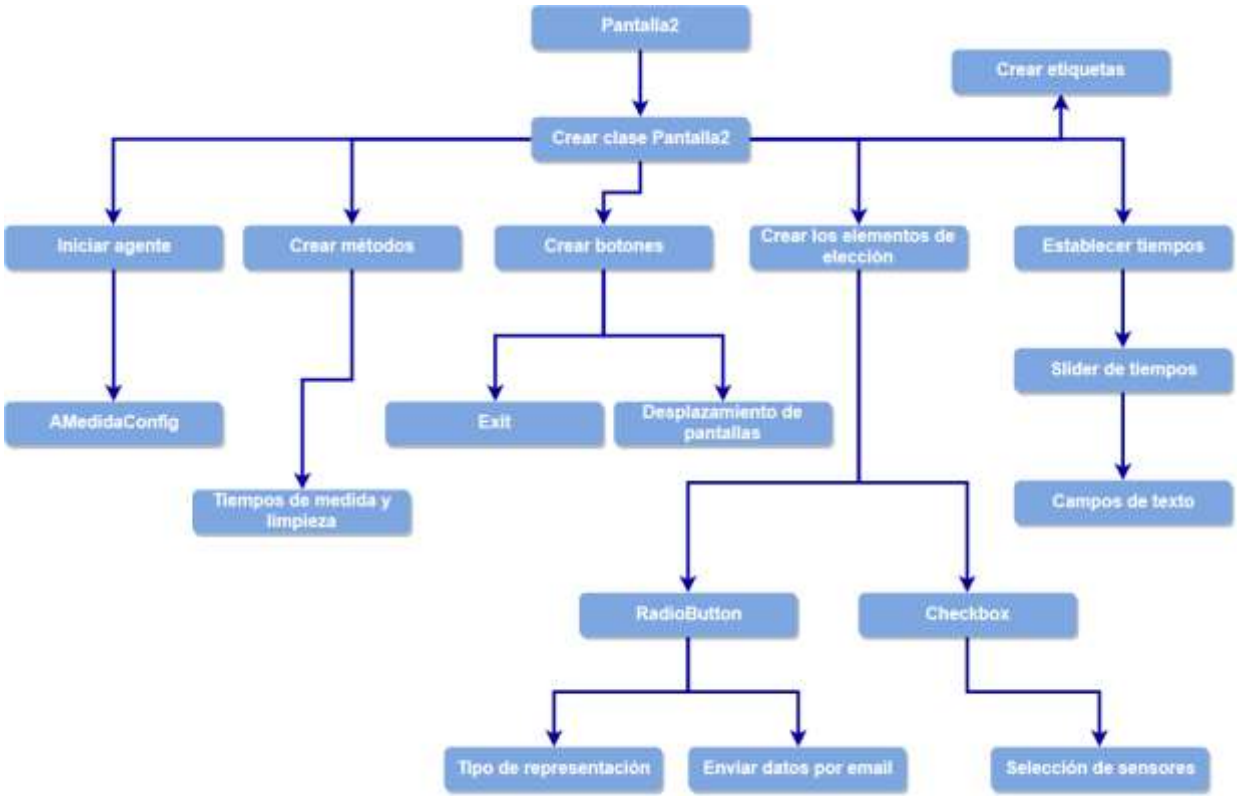


Ilustración 9 Esquema Pantalla2

La clase Pantalla2 pertenece al agente AMedidaConfig, por cosiguiente lo primero es inicializarlo.

```
public class Pantalla2 extends javax.swing.JFrame {

    Agent AMedidaConfig;

    /**
     * Creates new form Pantalla2
     */
    public Pantalla2(Agent agent) {
        initComponents();
        this.AMedidaConfig = agent;
    }
```

Figura 2.51 Asignación de pertenencia de Pantalla2 a AMedidaConfig

Los dos únicos métodos creados tienen la función de guardar el valor de tiempo que marca el campo de texto. Estos valores de tiempo son posteriormente enviados por mensaje al agente de AComunicacionI2C para que lo incluya en su flujo de control.

```
public void getTimeM(double MTimeValue){
    MTimeValue = Double.parseDouble(jTextField3.getText());
}

public void getTimeC(double CTimeValue){
    CTimeValue = Double.parseDouble(jTextField4.getText());
}
```

Figura 2.52 Métodos para la obtención de los tiempos de medida y limpieza en sus respectivas variables

En esta pantalla tenemos tres botones de nuevo. Dos de desplazamiento de pantalla y uno de 'Exit'. El desplazamiento de pantalla hacia Pantalla1 es normal, igual que en el caso anterior. Sin embargo el desplazamiento de pantalla hacia Pantalla3 tiene la función no solo de cambiar de pantalla, sino de también de pasar por mensaje al agente primario AComunicacionI2C los datos de los tiempos establecidos y obtenidos con los métodos anteriores.

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {

        AgentControl.sendMessage(AMedidaConfig, Agents.getAgentVisMedida(), new Message.PropertyValue(AVisMedidas.MENSAJE1));

        double MTimeValue = Double.parseDouble(jTextField3.getText());
        AgentControl.sendMessage(AMedidaConfig, Agents.getAgentComunicacionI2C(), new Message.PropertyValue(AComunicacionI2C.MTime,MTimeValue));

        double CTimeValue = Double.parseDouble(jTextField4.getText());
        AgentControl.sendMessage(AMedidaConfig, Agents.getAgentComunicacionI2C(), new Message.PropertyValue(AComunicacionI2C.CTime,CTimeValue));

    } catch (Exception e) {
        System.out.println("Excepcion en pantalla1: "+ e);
    }
}
```

Figura 2.53 Botón de desplazamiento de pantalla y envío de variables por mensaje a otro agente

En la figura superior se puede ver la estructura del envío de mensaje de nuevo. En este caso además de enviar un mensaje de un agente a otro, se envía una variable acoplada al mensaje. De esta manera en la recepción del mensaje por parte del agente receptor, se puede utilizar la variable en la parte del código que se ejecuta tras el 'case'.

El botón de 'Exit' tiene dos funciones, salir de la aplicación y apagar el relé mecánico que conecta la alimentación de la cámara de sensores. En la aplicación hay tres botones para salir y en todos ellos se desactiva la alimentación de los sensores. Al iniciar la aplicación, es el agente primario el que activa la alimentación de la placa de sensores. Esta activación y desactivación se hace a través del multiplexor MCP23008, como ya vimos en su capítulo correspondiente.

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    byte reles=0x00;
    I2CBus busMCP;
    try {
        busMCP = I2CFactory.getInstance(I2CBus.BUS_1);
        // Get I2C device, MCP23008 I2C address is 0x20(32)
        I2CDevice deviceMCP = busMCP.getDevice(0x27);
        deviceMCP.write(0x00, (byte)0x00);

        reles = (byte)(0x00);
        reles = (byte)(reles & ~(1<<0));  // DESACTIVAR
        deviceMCP.write(0x09, (byte)reles);

    } catch (I2CFactory.UnsupportedBusNumberException ex) {
        Logger.getLogger(AComunicacionI2C.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(AComunicacionI2C.class.getName()).log(Level.SEVERE, null, ex);
     }
    System.exit(0);
}
```

Figura 2.54 Botón de Exit para salir de la aplicación y desactivar la alimentación de los sensores

Los elementos de elección se emplean para hacer decidir al usuario sobre tres temas:

- Si quiere obtener una representación de los datos de medida referenciados respecto a los datos de limpieza o si los quiere absolutos.
- Si desea interaccionar a través de internet con la nariz y enviarse a sí mismo los datos por correo.
- Y por último, qué datos de sensores quiere que la nariz recoja en el proceso de medida.

Para ello se han utilizado los elementos de Java ‘jRadioButton’ y ‘jCheckBox’.

```
private void jCheckBox1ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        boolean status = jCheckBox1.isSelected();
        AgentControl.sendMessage(AMedidaConfig, Agents.getAgentComunicacionI2C(), new Message.PropertyValue(AComunicacionI2C.STATUS_S1,status));
        AgentControl.sendMessage(AMedidaConfig, Agents.getAgentGRadial(), new Message.PropertyValue(AGRadial.STATUS_S1,status));
        AgentControl.sendMessage(AMedidaConfig, Agents.getAgentSuperuser(), new Message.PropertyValue(ASuperuser.CHANGE_STATUS_S1,status));

    } catch (IterativeThread.IterativeThreadException ex) {
        Logger.getLogger(Pantalla2.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Figura 2.55 Elemento para seleccionar los sensores

La figura superior es un ejemplo para la selección del sensor 1, donde se puede ver el envío de mensaje a otros agentes con el valor de la variable booleana ‘status’ asociada. Esta era la variable que hemos comentado en el punto 2.3.2 respecto al estado de los campos de texto en Pantalla1.

La selección del tipo de dato a representar o el envío por correo, se hace del mismo modo. En el caso de envío por correo, al pulsar el ‘jRadioButton’ se realizan dos

acciones. Comprobar el estado de la variable y enviar el mensaje que haga saltar la pantalla con el teclado. Este teclado nos sirve para escribir en la pantalla táctil nuestro correo y enviarlo de vuelta al agente primario.

```
private void jRadioButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        boolean status = jRadioButton1.isSelected();
        AgentControl.sendMessage(AMedidaConfig, Agents.getAgentComunicacionI2C(), new Message.PropertyValue(AComunicacionI2C.EMAILORNOT,status));
        AgentControl.sendMessage(AMedidaConfig, Agents.getAgentSuperuser(), new Message.PropertyValue(ASuperuser.MENSAJE3));
    } catch (IterativeThread.IterativeThreadException ex) {
        Logger.getLogger(Pantalla2.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Figura 2.56 RadioButton para la opción de enviar la medida por correo

Figura 2.57 Teclado táctil

Una vez que se introduce el correo en el campo de texto de la parte superior, el botón de ‘OK’ envía un mensaje al agente AComunicacionI2C con la variable asociada de la dirección de correo.

Por último, el establecimiento del tiempo de medida y limpieza se realiza deslizando las barras en la parte central derecha de la pantalla (Ver Figura 2.61). El valor de tiempo que establezcamos deslizando la barra, se reflejará en los campos de texto que hay justo debajo. Los dos métodos que se vieron al principio de este punto cogen los datos de tiempo de estos dos campos de texto. Los datos de tiempo están expresados en segundos, con un máximo de 100 segundos, aunque podría aumentarse hasta 120. La relación entre el ‘jSlider’ y el ‘jTextField’ se hace con la opción de Binding del propio campo de texto.

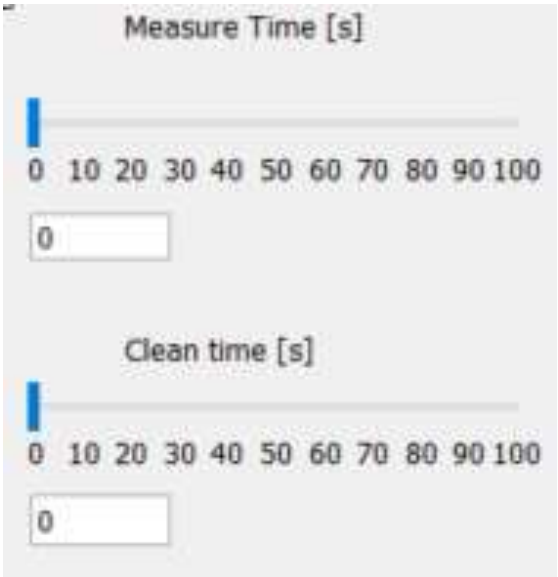


Figura 2.58 Sliders para establecer el tiempo de medida y limpieza

2.3.4. Pantalla de Visualización de medidas I

Para la visualización de la medida hay dos pantallas en la interfaz de la nariz electrónica. Se representan los datos de dos formas distintas para dar toda la información posible en vivo al usuario. En esta pantalla se visualiza la primera representación de datos, que corresponde con una evolución temporal de las curvas de los sensores al paso del flujo de aire de las muestras de aceite. Los valores de la curva aumentarán o disminuirán en función de los volátiles que sean detectados por los sensores.

Además esta pantalla tiene la función de interaccionar con el usuario de manera que a este se le permita influir en el flujo de la programación. Es decir, botones para empezar, resetear, parar o guardar la medida realizada. Tiene además la monitorización de la temperatura y la humedad en el interior de la cámara de sensores.

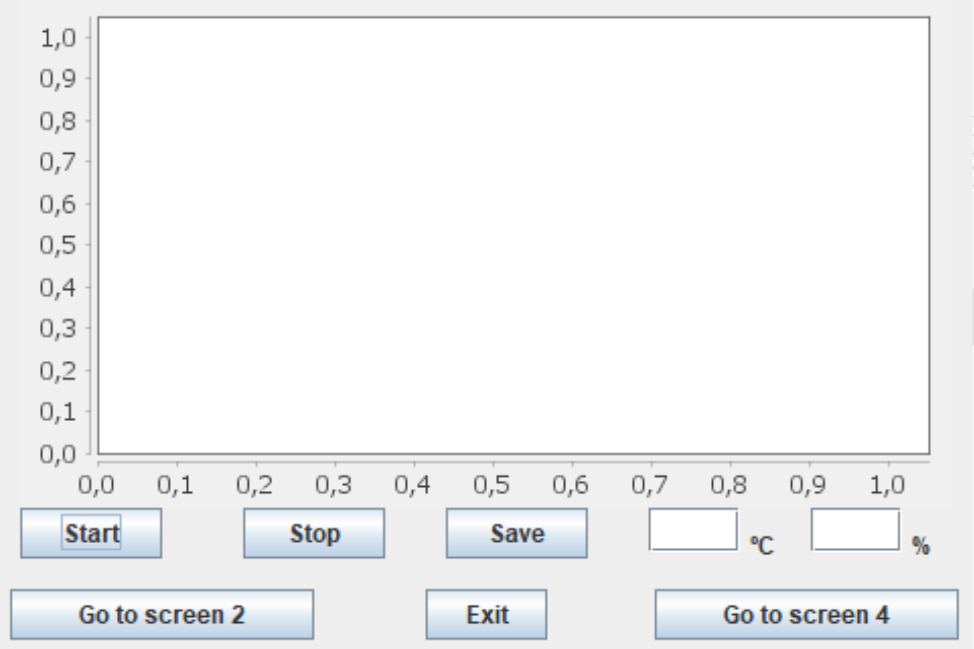


Figura 2.59 Pantalla de visualización de la medida I

En esta captura de la interfaz se aprecian los botones y funciones citados en el párrafo superior. La parte principal de la pantalla es un área en la que se hace la representación gráfica de las curvas de los sensores. El eje x es el tiempo de la medida en segundos y el eje y es el valor de resistencia del sensor. Conforme va aumentando el tiempo en el eje x, se va refrescando la gráfica de izquierda a derecha hasta llegar al momento final de medida.

Los tres botones de la parte inferior son los mencionados anteriormente que permiten controlar el flujo de la programación. El botón 'Start' sirve para iniciar la medida una vez configurada y para configurar y realizar una nueva medida cuando se haya realizado alguna otra antes. El botón de 'Stop' paraliza la medida en el instante en el que se pulse. El botón de 'Save' guarda la medida en un archivo .csv y si se ha seleccionado la opción de enviar por email de la pantalla anterior, se envía el email.

En primer lugar explicaremos el agente que controla esta pantalla y en segundo lugar la clase asociada a él.

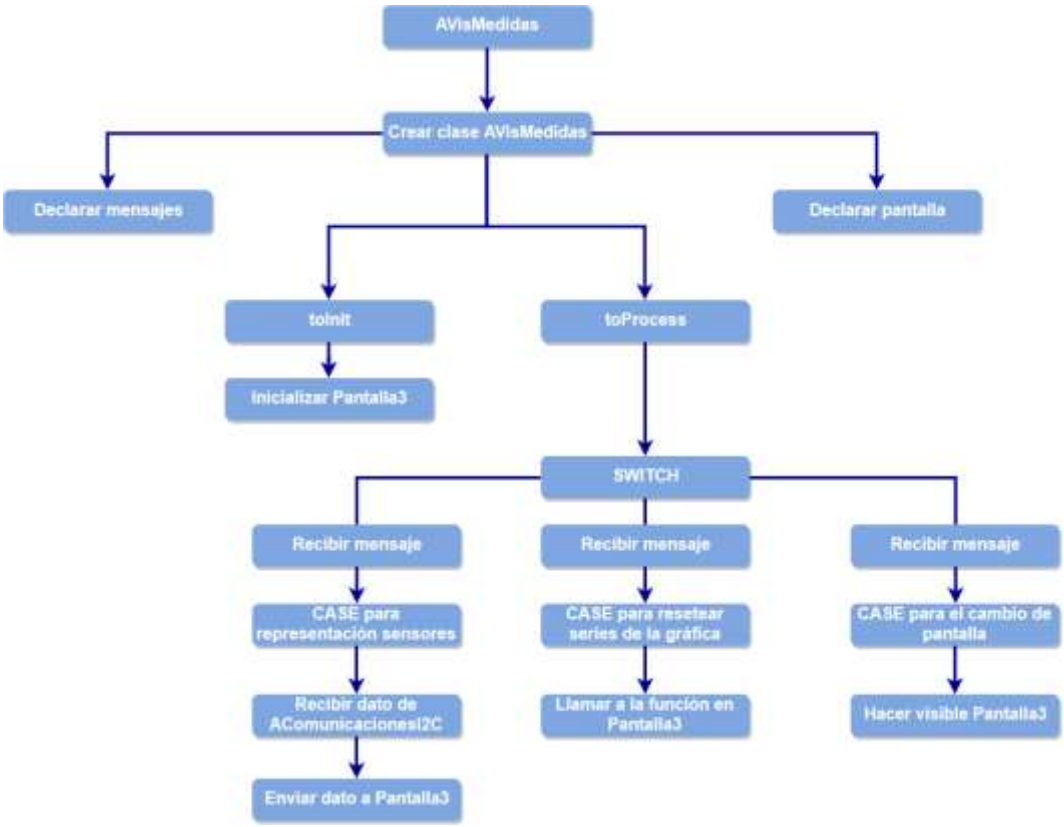


Ilustración 10 Esquema AVisMedidas

Tras la declaración de mensajes necesarios y la pantalla, en el 'toInit' en este caso solo se inicializa la clase Pantalla3. En el 'toProcess' se hacen varias tareas. La primera la recepción de mensajes de AComunicacionI2C con los datos de cada sensor. Después de la recepción se pasan a Pantalla3 a través de un método.

```
case AComunicacionI2C.DATOSGS1:
    System.out.println("Graficar sensor 1");
    miPantalla.setSerieS1((double)m.data.value);
      miPantalla.repaint();
    break;
```

Figura 2.60 Recepción y envío de mensaje con datos de un sensor para su representación

La segunda, el reseteo de la gráfica una vez que se haya realizado una medida y se quiera volver a hacer otra. La gráfica queda limpia de curvas de sensores para la siguiente medida.

```
case AComunicacionI2C.RESETSERIES:
    miPantalla.clearSeries();
    break;
```

Figura 2.61 Reseteo de las curvas de la gráfica

Y por último los 'case' para el desplazamiento de la pantalla como en todas las pantallas anteriores.

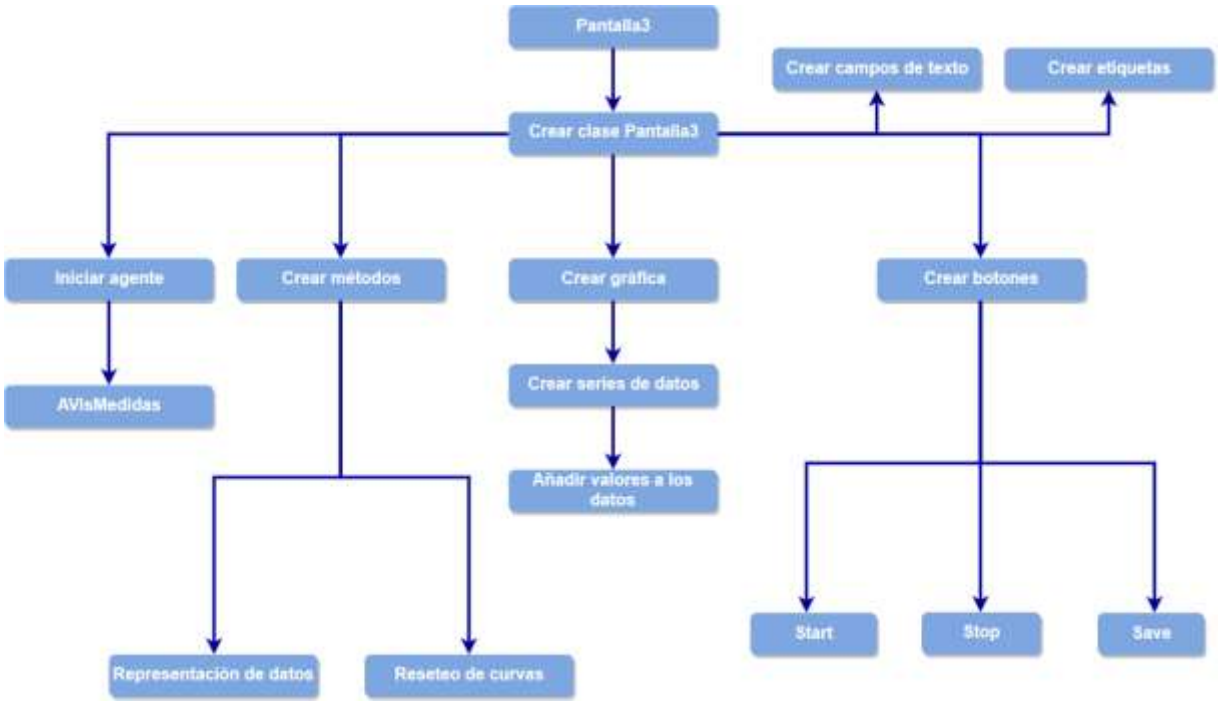


Ilustración 11 Esquema Pantalla3

Igual que en las otras pantallas, se inicializa el agente correspondiente, se crean los métodos y los elementos de la interfaz.

En Pantalla3 se parte por la creación de la gráfica central donde se representan las curvas. Para ello se utiliza una librería especial citada en el punto 2.1.2.2, JFreeChart. La librería puede implementar distintos tipos de gráficos, las posibilidades son muchas. Para cumplir con el diseño, se ha usado un tipo de series XY. Las series XY son fáciles de usar y se crean siempre a partir de una pareja de valores, en este caso valor de sensor y valor de tiempo.

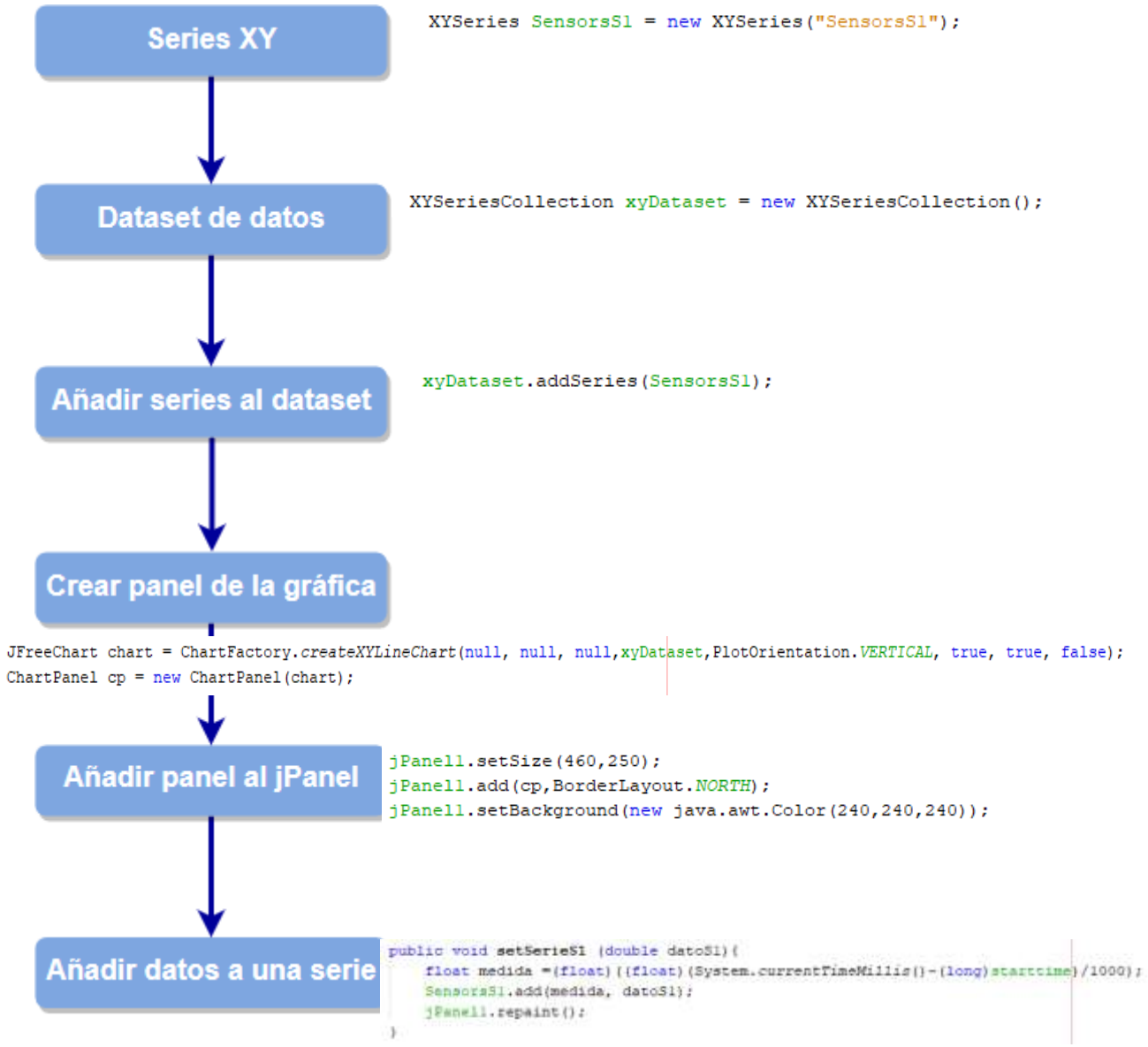


Ilustración 12 Crear gráfica XY

El último paso de añadir datos a una serie es un método que se encarga precisamente de añadir los datos del sensor y el tiempo del modo que se ve en la figura superior junto al recuadro azul.

Los botones inferiores internamente no son los que controlan el flujo realmente. Los botones solo envían un mensaje al agente primario, para que este en cada ‘case’ haga lo que corresponde.

2.3.5. Pantalla de Visualización de medidas II

La segunda pantalla para la visualización de la medida es una representación característica de la huella aromática del aceite (Ver Figura 2.65). Su función es la de proporcionar otra perspectiva de la medida, más intuitiva y que aporta más información que la gráfica XY evolutiva. Tiene además un listado histórico de medidas del que se puede seleccionar una de ellas y ver de nuevo la huella que ha dado una determinada muestra de aceite. Quizás sea la pantalla más llamativa e interesante para el usuario, ya que queda claro que la forma de la huella puede ser característica según su morfología.

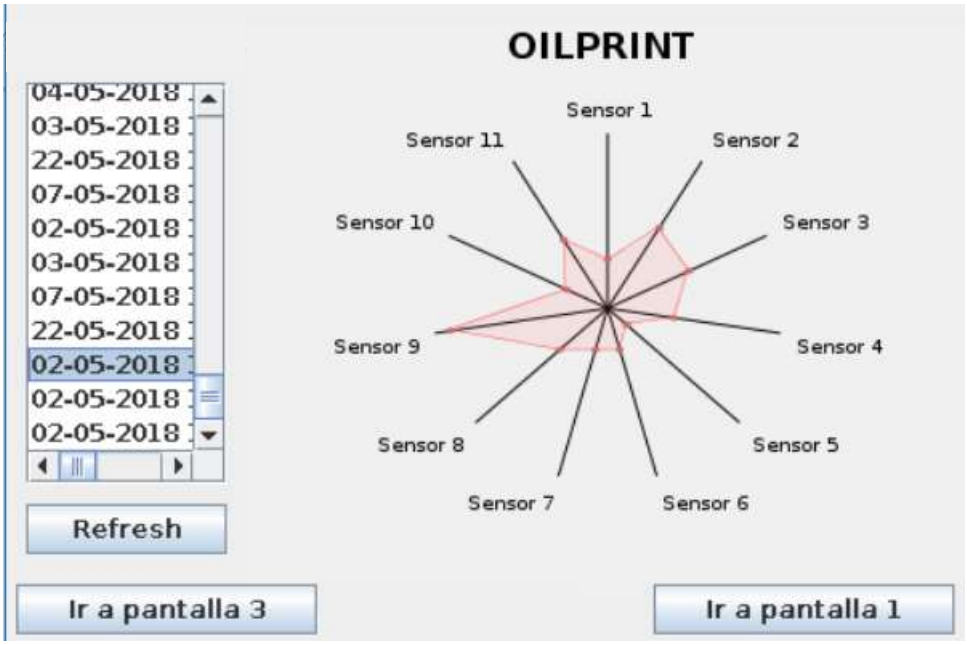


Figura 2.62 Pantalla de visualización de la medida II

Por ejemplo, en la figura superior está seleccionada una medida del día 2 de mayo de 2018 y la muestra presenta los picos determinados en cada sensor. Pantalla4 es la última clase que tiene interfaz en la nariz electrónica, aunque no se descarta que en el futuro se puedan añadir nuevas pantallas y nuevas funciones.

La clase Pantalla4 pertenece al agente AGRadial (Ver Ilustración 13) y tiene exactamente la misma estructura que los módulos anteriores. El tipo de representación de datos afecta del mismo modo a la huella, puede representarse datos referenciados o absolutos. Los datos absolutos en esta representación no tienen

mucho sentido porque habría tanta diferencia entre los picos máximo y mínimo que apenas podría verse una huella representativa en la gráfica.

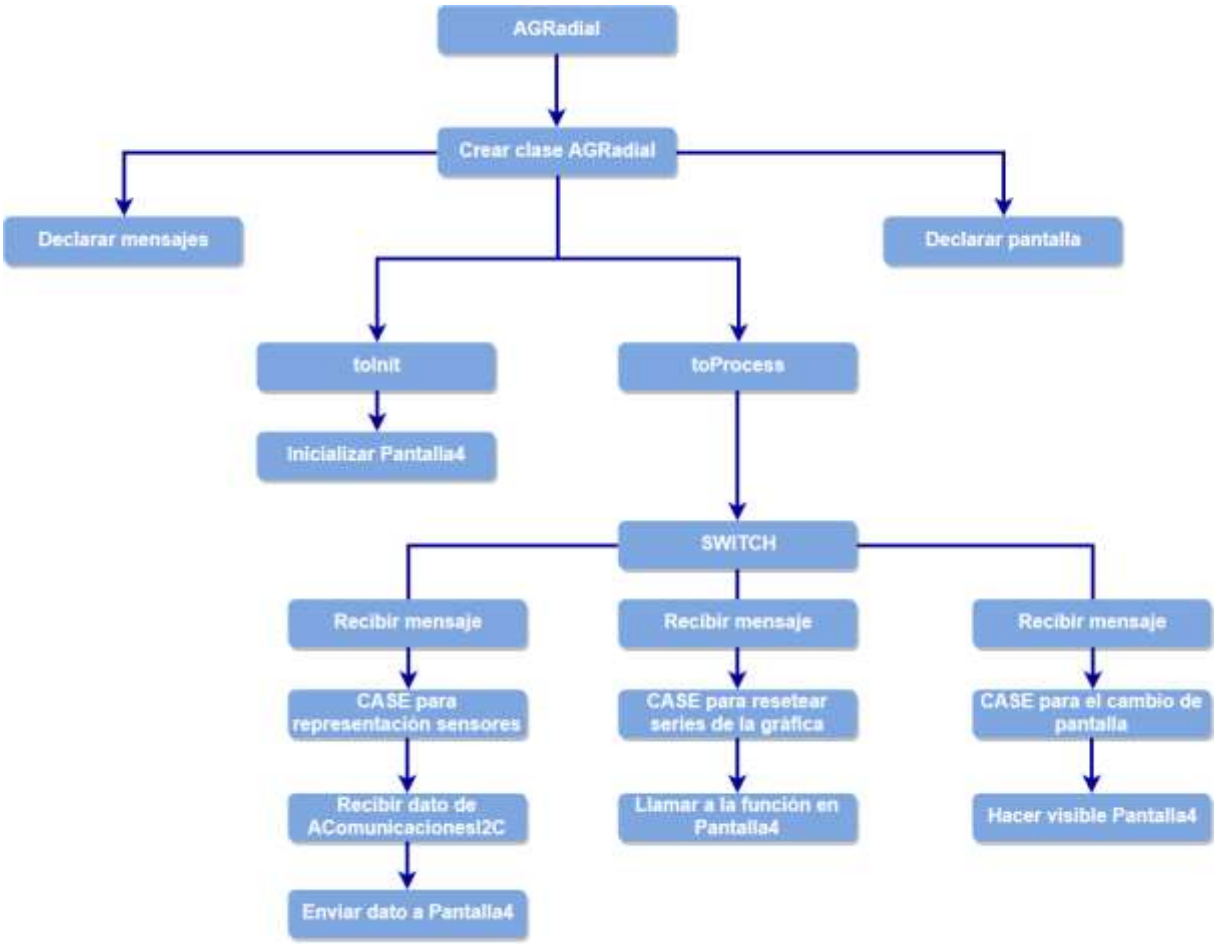


Ilustración 13 Esquema AGRadial

El esquema del agente AGRadial es exactamente igual que el agente AVisMedidas. La filosofía es la misma: recoger datos de sensores, pasarlos mediante función a su clase asociada y enviar otro mensaje cuando haya que limpiar la gráfica de la huella para una nueva medida.

```
case AComunicacionI2C.DATOSSGS11:
    ValueS11=((double)m.data.value);
    miPantalla.setValueS11(ValueS11);
    break;
case AComunicacionI2C.RESETSPIDER:
    miPantalla.clearSeriesSpider();
    break;
```

Figura 2.63 Paso de datos por función y llamar a función de reseteo de gráfica

La estructura de la clase Pantalla4 es muy parecida a la clase Pantalla4, exceptuando la función del histórico de medidas que incluye Pantalla4.

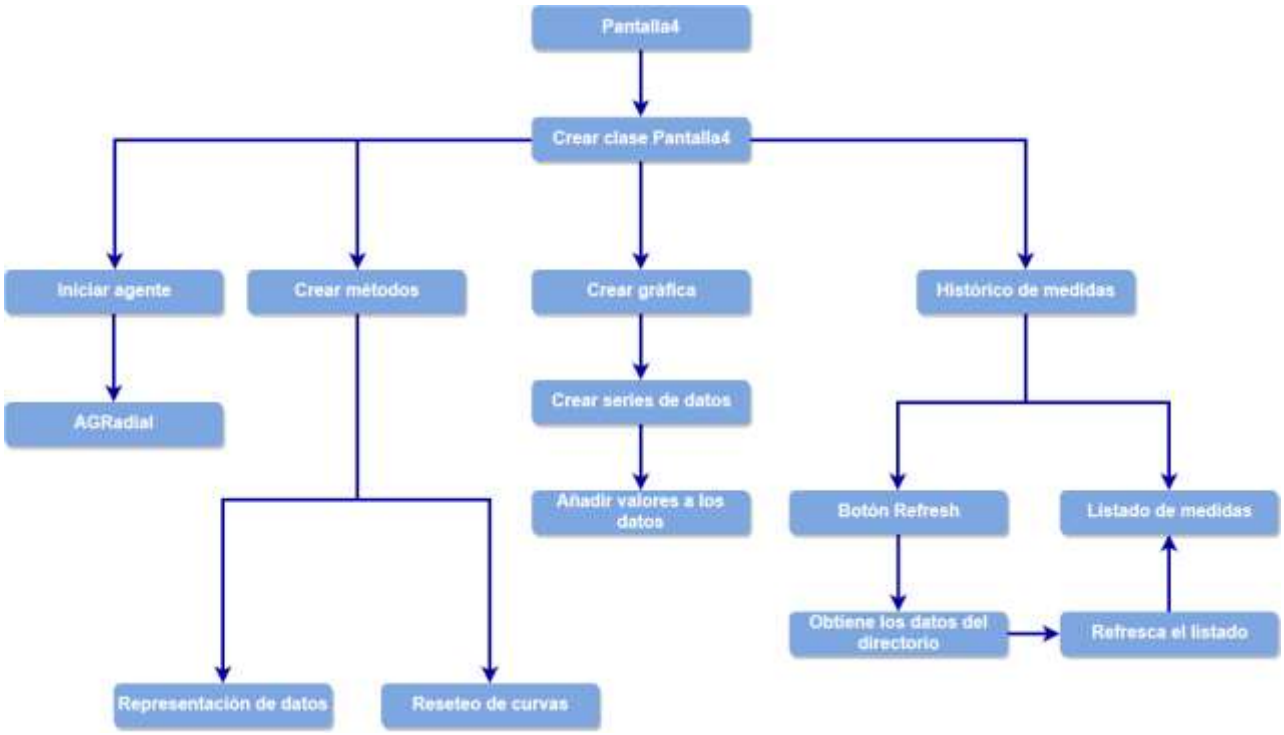


Ilustración 14 Esquema Pantalla4

Aunque la estructura del esquema sea idéntica, el tipo de gráfica utilizado también varía. En este caso, haciendo uso de nuevo de la librería 'JFreeChart', se crea una gráfica llamada 'SpiderWebPlot', que da el resultado de la anterior Figura 2.65.

Los parámetros de entrada que necesita la huella son dos: el valor de la medida del sensor y el eje en el que se coloca. Al final la huella se divide en tantos ejes como se le indique, es decir, tantos valores como se añadan a la gráfica.

```
SpiderWebPlot plot = new SpiderWebPlot(dataset);
plot.setMaxValue(4.00);
JFreeChart chart = new JFreeChart("OILPRINT ", plot);
ChartPanel cp = new ChartPanel(chart);
plot = (SpiderWebPlot) cp.getChart().getPlot();

dataset = (DefaultCategoryDataset) plot.getDataset();

jPanel1.setSize(360,280);
jPanel1.add(cp,BorderLayout.NORTH);
jPanel1.setBackground(new java.awt.Color(240,240,240));
```

Figura 2.64 Creación de gráfica SpiderWebPlot

La última función que vamos a explicar de las pantallas de la nariz electrónica es el listado de medidas que se puede ver en la parte izquierda de Pantalla4. El botón 'Request' desencadena una acción en el flujo que permite a Java entrar en los directorios de la Raspberry y buscar en el directorio indicado las medidas realizadas anteriormente y evidentemente, las que se hayan guardado. Al refrescar la lista, el código recorre todo el directorio archivo por archivo leyendo su nombre. Si el nombre del archivo ya está cargado en la lista, no se añade de nuevo a la lista para no tener repetidos.

Una vez que las medidas están en el listado, es posible pulsar sobre ellas y automáticamente el archivo se carga en la gráfica tipo huella. Para cargar correctamente los datos desde el archivo, se hace un procesado previo y se seleccionan los datos a representar, convirtiéndolos siempre a datos referenciados. El elemento de Java para implementar esta funcionalidad es un 'jList', en el que se programa un evento para cuando uno de los elementos de la lista haya sido pulsado.

```
String linea [] = new String [15];
String lineaC [] = new String [15];
BufferedReader rd = null;
  try {
    rd = new BufferedReader(new FileReader (ruta+jList1.getSelectedValue().toString()));
    String line = "";
    String line1 = "";
    String lineIntermedio = "";
    String lineC = "";
    while ((line = rd.readLine()) != null) {
      line1=line;
      line = rd.readLine();
      if((line1.indexOf("clean")!=-1)){
          System.out.println("Estoy dentro del primer if");
          lineIntermedio=line1;
          if((line.indexOf("meas")!=-1)){
            System.out.println("Estoy dentro del segundo if");
          }
      }
      System.out.println(line);
      lineC=lineIntermedio;
    }
    System.out.println(line1);
    System.out.println(lineC);
    linea = line1.split(";");
    lineaC = lineC.split(";");
```

Figura 2.65 Selección del elemento de la lista y búsqueda en el fichero de datos

2.3.6. Agente de Comunicaciones

Después de haber explicado todas las pantallas de la nariz electrónica una por una, se podría llegar fácilmente a la conclusión de que todos los agentes y por tanto sus clases también, dependen del agente AComunicacionI2C o agente primario. Este agente es el que realmente tiene el control sobre todos los demás. Esto es debido a que los datos de los sensores son obtenidos directamente por él y él mismo se encarga de enviarlos a los demás agentes.

Como todos los demás agentes, AComunicacionI2C necesita declarar mensajes para poder comunicarse. Solo que éste necesita un total de 82 mensajes para cubrir todos los 'case' que necesita. La gran cantidad de opciones dentro del flujo de control se debe en parte a que hay 12 sensores en la matriz, por tanto hay operaciones que se repiten obligatoriamente 12 veces.

En el 'toInit' solo se lleva a cabo una acción: la activación del relé que alimenta la placa de sensores. Se considera el encendido de la nariz electrónica, ya que empieza a funcionar la placa principal y se conectan los sensores al bus de datos. Anteriormente hemos visto que este relé se desactiva mediante botones de 'Exit' en algunas de las pantallas de la interfaz.

En el 'toProcess' lo primero que se hace es crear los buses I2C de los elementos, los tres convertidores, el sensor SHT31 y el multiplexor.

```
public synchronized void toProcess() throws IterativeThreadException, InterruptedException {
    try {
        // Create I2CBus CONVERTIDOR 1
        I2CBus busC1 = I2CFactory.getInstance(I2CBus.BUS_1);
        I2CDevice deviceC1 = busC1.getDevice(0x48);

        // Create I2CBus CONVERTIDOR 2
        I2CBus busC2 = I2CFactory.getInstance(I2CBus.BUS_1);
        I2CDevice deviceC2= busC2.getDevice(0x49);


        // Create I2CBus CONVERTIDOR 3
        I2CBus busC3 = I2CFactory.getInstance(I2CBus.BUS_1);
        I2CDevice deviceC3= busC3.getDevice(0x4B);

        // Create I2CBus
        I2CBus busSHT = I2CFactory.getInstance(I2CBus.BUS_1);
        // Get I2C device, SHT31 I2C address is 0x44(68)
        I2CDevice deviceSHT = busSHT.getDevice(0x44);

           Create I2C bus
        I2CBus busMCP = I2CFactory.getInstance(I2CBus.BUS_1);
        // Get I2C device, MCP23008 I2C address is 0x20(32)
        I2CDevice deviceMCP = busMCP.getDevice(0x27);
```

Figura 2.66 Creación del bus I2C y sus elementos

A continuación comienzan todos los ‘case’ que recogen las variables de los sensores y las variables que las pantallas de la interfaz gráfica vistas anteriormente han ido aportando al flujo del agente primario. Como el esquema de AComunicacionI2C sería muy amplio, detallaremos por puntos las acciones que realiza:

- Recibe y utiliza los datos del tiempo de medida y limpieza tras configurar la medida en Pantalla2.
- Recibe y utiliza el email aportado por el usuario en Pantalla2.
- Calcula los datos de los sensores desde los canales de cada convertidor en absoluto y en referenciado.
- Recibe la orden de la botonera del control de flujo en Pantalla3 y ejecuta el código en consecuencia.
- Calcula el tiempo de comienzo y final de la medida para el eje de tiempos de la representación gráfica.
- Activa y desactiva las válvulas que direccionan el flujo de aire según el ciclo sea de medida o de limpieza.
- Guarda los datos de cada medida realizada en un archivo .csv en el directorio indicado de la Raspberry.
- Envía por email un correo con un archivo adjunto de la medida que se acaba de realizar.

Todo esto lo hace siguiendo las técnicas que se han utilizado en los otros agentes. El uso de la programación basada en agentes permite que todas las tareas comentadas se realicen en el momento en que se soliciten. El código resultante en este agente es, por tanto, muy extenso comparado con los demás agentes y explicar de nuevo cómo se envían mensajes y cómo funcionan por dentro los agentes no nos parece necesario. Sí es cierto que el envío por email es la única parte que tiene estructura diferente y resulta llamativa por el acercamiento al IoT que se consigue en esta primera versión de la nariz electrónica.

Se trata de utilizar el servidor de Google SMTP, su puerto y las credenciales de autenticación necesarias para acceder a una sesión de Gmail. Se creó una cuenta Gmail a la nariz electrónica del GRAV y es desde esta cuenta donde se envían los correos al usuario.

```
if (flagemail){
 ////////////////// SEND EMAIL ///////////////////////
    try{
        // Esto es lo que va delante de @gmail.com en tu cuenta de correo. Es el remitente también.
        System.out.println("ENVIANDO EMAIL");
        String remitente [redacted]v";  //Para la dirección nomcuenta@gmail.com
        String clave = [redacted];
        InternetAddress toAddress=null;
        toAddress = new InternetAddress(email);
        Properties props = System.getProperties();
        props.put("mail.smtp.host", "smtp.gmail.com");  //El servidor SMTP de Google
        props.put("mail.smtp.user", remitente);
        props.put("mail.smtp.clave", "nariz1234");    //La clave de la cuenta
        props.put("mail.smtp.auth", [redacted];    //Usar autenticación mediante usuario y clave
        props.put("mail.smtp.starttls.enable", "true"); //Para conectar de manera segura al servidor SMTP
        props.put("mail.smtp.port", "587"); //El puerto SMTP seguro de Google

        Session session = Session.getDefaultInstance(props);
        MimeMessage message = new MimeMessage(session);

        try {
            message.setFrom(new InternetAddress(remitente));
            message.addRecipient(javax.mail.Message.RecipientType.TO, toAddress);   //Se podrian añadir varios de la misma manera
            message.setSubject("MEDIDA REALIZADA");


            Multipart multipart = new MimeMultipart();

            MimeBodyPart textBodyPart = new MimeBodyPart();
            textBodyPart.setText(NombreFichero);

            MimeBodyPart attachmentBodyPart = new MimeBodyPart();
            DataSource source = new FileDataSource(ruta+NombreFichero+".csv");
            attachmentBodyPart.setDataHandler(new DataHandler(source));
            attachmentBodyPart.setFileName(NombreFichero+".csv");

            multipart.addBodyPart(textBodyPart);
            multipart.addBodyPart(attachmentBodyPart);

            message.setContent(multipart);

            Transport transport = session.getTransport("smtp");
            transport.connect("smtp.gmail.com", remitente, clave);
            transport.sendMessage(message, message.getAllRecipients());
            transport.close();
            System.out.println("EMAIL ENVIADO");
        }
```

Figura 2.67 Fragmento del código donde de envía el email

3. RESULTADOS

El trabajo realizado durante el proceso de desarrollo del proyecto ha dado sus frutos. El resultado es el esperado y positivo. Se han cumplido con los objetivos marcados en el capítulo de introducción, en el punto 1.3., aunque, por supuesto, hay posibilidades de mejora como se verá en el siguiente capítulo.

El software de la nariz electrónica se terminó a finales de abril, por lo cual hubo tiempo suficiente para hacer pruebas y empezar a pasar muestras de aceite catalogadas por nuestra nariz. La parte hardware estaba para esa fecha también terminada y preparada para empezar a funcionar, al igual que la parte mecánica que se fabricó en 3D y mecanizó piezas de la cámara de sensores, soporte de pantalla, etc.

Es cierto que el resultado obtenido no se considera la nariz electrónica industrial que puede llegar a ser en un futuro cercano. Actualmente tenemos un prototipo terminado listo para funcionar y comenzar el proceso de validación del proyecto. Cualquier proyecto que se acometa de estas características necesita un periodo de validación suficientemente largo como para asegurar que el concepto es válido y puede desempeñar la labor para la que ha sido confeccionado. Este periodo para la nariz se extenderá hasta finales de agosto o principios de septiembre. El objetivo es preparar la nariz para la siguiente campaña de recogida de aceituna y obtención de aceite. Esto no quiere decir que la nariz para esa época sea un producto industrial consolidado, casi con total seguridad seguiría siendo una máquina en pruebas dentro de las almazaras las cuales tendrían la posibilidad de valorar si realmente la nariz electrónica puede distinguir la calidad del aceite producido en proceso.

Por este motivo, hablar de resultados aún es complicado y habría que esperar a siguientes fases del proyecto. Para la fase del proyecto correspondiente a este trabajo de fin de grado y el de los otros dos compañeros, el resultado es muy positivo. Mostraremos el software obtenido en esta parte del proyecto, el prototipo final que se ha conseguido uniendo las tres partes proyectadas y por último el proceso de las primeras muestras de aceite que se han pasado por la nariz, intentando que sirvan de clasificadoras de la calidad mediante técnicas de análisis de datos avanzadas.

3.1. Software desarrollado

Se ha conseguido la integración completa de una interfaz en Java para la nariz electrónica utilizando como soporte una Raspberry Pi. La interfaz es capaz de interactuar con el usuario en:

- Configuración de la medida individual para cada muestra.
- Establecer tiempos de medida y limpieza.
- Elegir qué sensores se quieren medir.
- Seleccionar la opción de enviar la medida por correo electrónico.
- Elegir el tipo de representación de datos.
- Supervisar el estado de los sensores.
- Visualizar la medida en una gráfica evolutiva.
- Guardar los datos en un archivo .csv para su posterior análisis.
- Visualizar la medida en una gráfica que aporta una huella característica de la calidad del aceite.
- Recuperar la huella de cualquier medida realizada y guardada en el histórico para visualizarla de nuevo.

Todas estas funciones se implementan en distintas pantallas por las que el usuario puede desplazarse fácilmente (Ver Figuras 3.1 a 3.6). Al ser una pantalla táctil, el uso del software de una nariz electrónica es muy similar a estar utilizando un teléfono móvil o tableta.

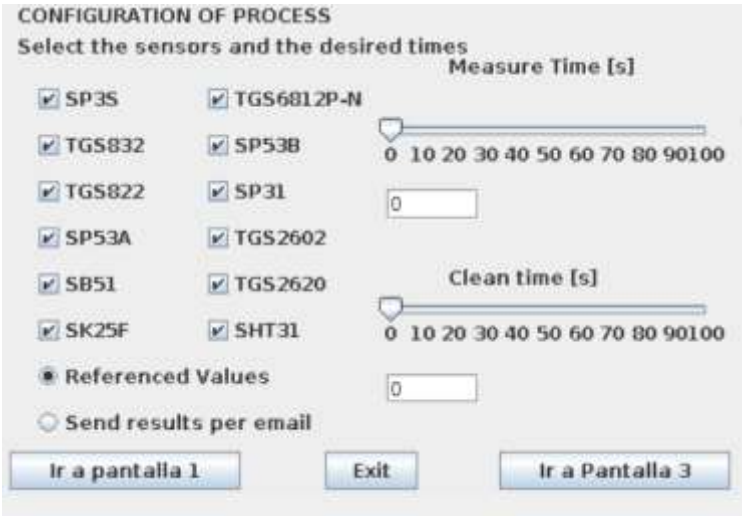


Figura 3.1 Pantalla de configuración de la medida

Figura 3.2 Pantalla de supervisión del estado de los sensores

Figura 3.3 Teclado

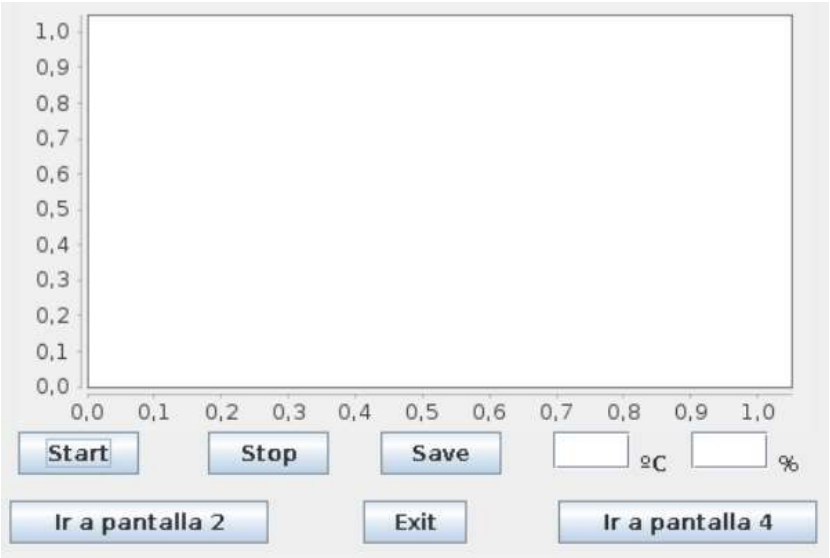


Figura 3.4 Pantalla de visualización de la medida

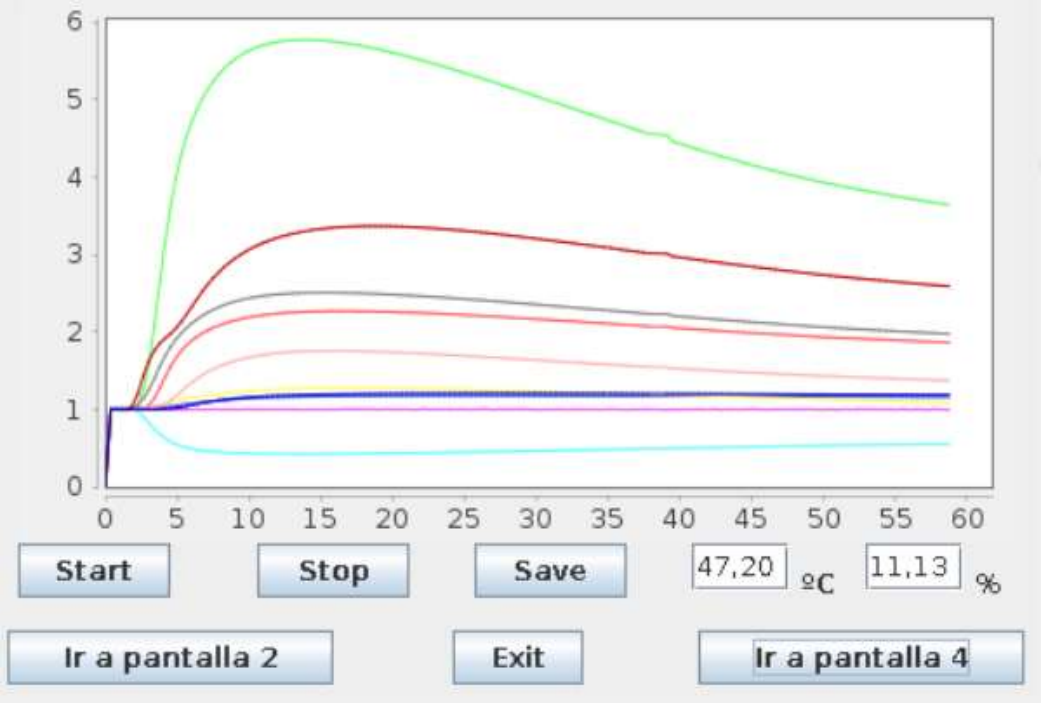


Figura 3.5 Pantalla de visualización de la medida en funcionamiento

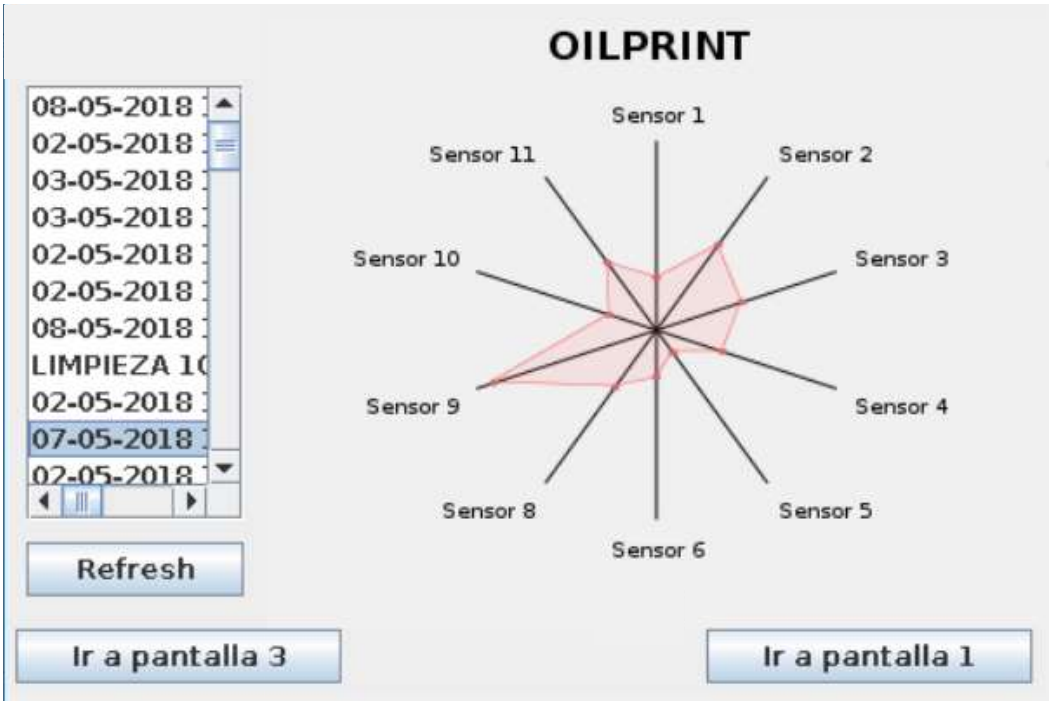


Figura 3.6 Pantalla de visualización de la huella sensorial

El capítulo 8, anexo de manual de usuario, estará dedicado a explicar a modo de guía cómo debe usarse la nariz electrónica.

3.2. Prototipo final

El prototipo está constituido por la unión de la mecánica, el hardware y el software de la nariz electrónica. Se ha tratado de integrar todo de forma compacta y eficiente para que ocupe el menor espacio posible y que su peso no sea elevado. En definitiva se ha intentado cumplir con las condiciones ergonómicas necesarias que requiere una máquina de uso industrial. La máquina además debe poder transportarse fácilmente porque no tiene por qué tener un lugar de funcionamiento fijo.

Visualmente también se ha procurado que la nariz resulte llamativa en el prototipo final, y no solo eso, sino que además sea fácil de explicar la electrónica que lleva dentro. De esta manera, el prototipo final se ha dejado abierto y acoplado sobre una placa en la que se pueden ver las placas PCB, la Raspberry, las electroválvulas, la bomba, etc. Lo más llamativo quizás sea la cámara de sensores, ya que ésta está cerrada y sellada con una tapadera de metacrilato transparente, de forma que se puede ver el interior de la cámara y los sensores utilizados. Para la explicación del proyecto, esta presentación facilitará la comprensión del funcionamiento de una nariz electrónica.

Una vez pasada la etapa del trabajo fin de grado, la nariz electrónica se integrará en una caja de metal para tomar forma de producto industrial. La caja ha sido mecanizada para tener acceso a los puertos periféricos de la Raspberry y para hacer de soporte de la pantalla táctil de 7'', con lo cual será sencillo trabajar con ella. La electrónica hardware irá dentro de la caja de forma que ocupe el menor espacio posible de manera que en futuras versiones pueda ser reducido el tamaño de la caja. Así mismo se ha tenido en cuenta en la parte mecánica las mejoras que ya se han pensado para la nariz electrónica, que tienen que ver con la ampliación de la nariz con una cámara supletoria contenedora de la muestra de aceite, que lo adapte a las condiciones óptimas de medida.

En las Figuras 3.7 a 3.16 que se incluyen a continuación se puede ver la diferencia entre el prototipo final preparado para presentar este trabajo y el producto industrial cerrado, aunque siga siendo un prototipo.

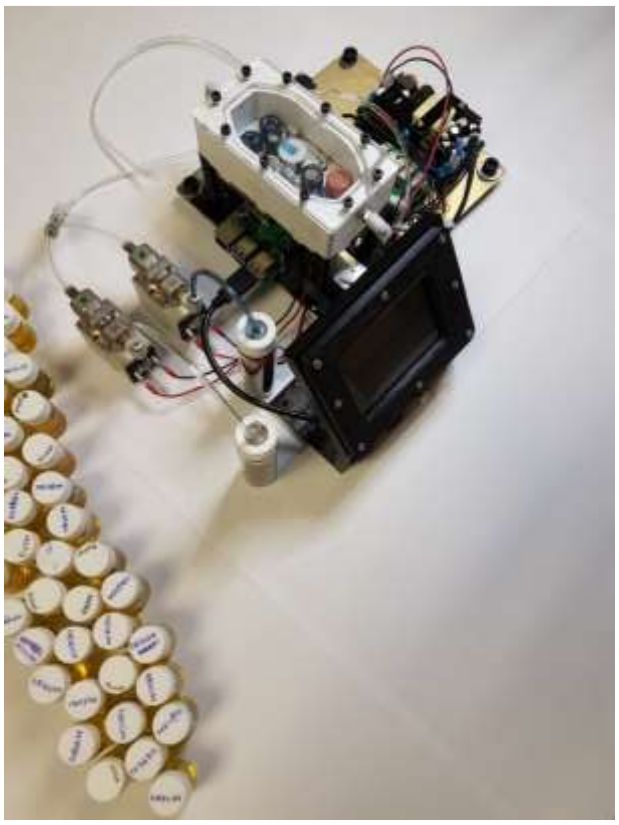


Figura 3.7 Prototipo 1

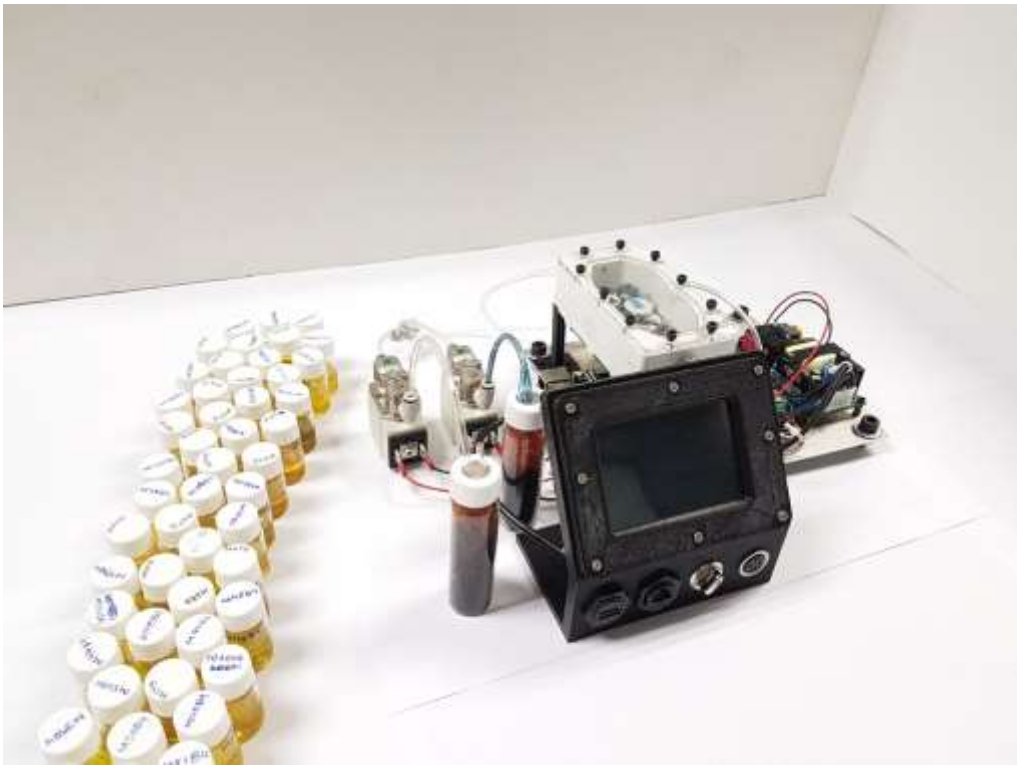


Figura 3.8 Prototipo 2

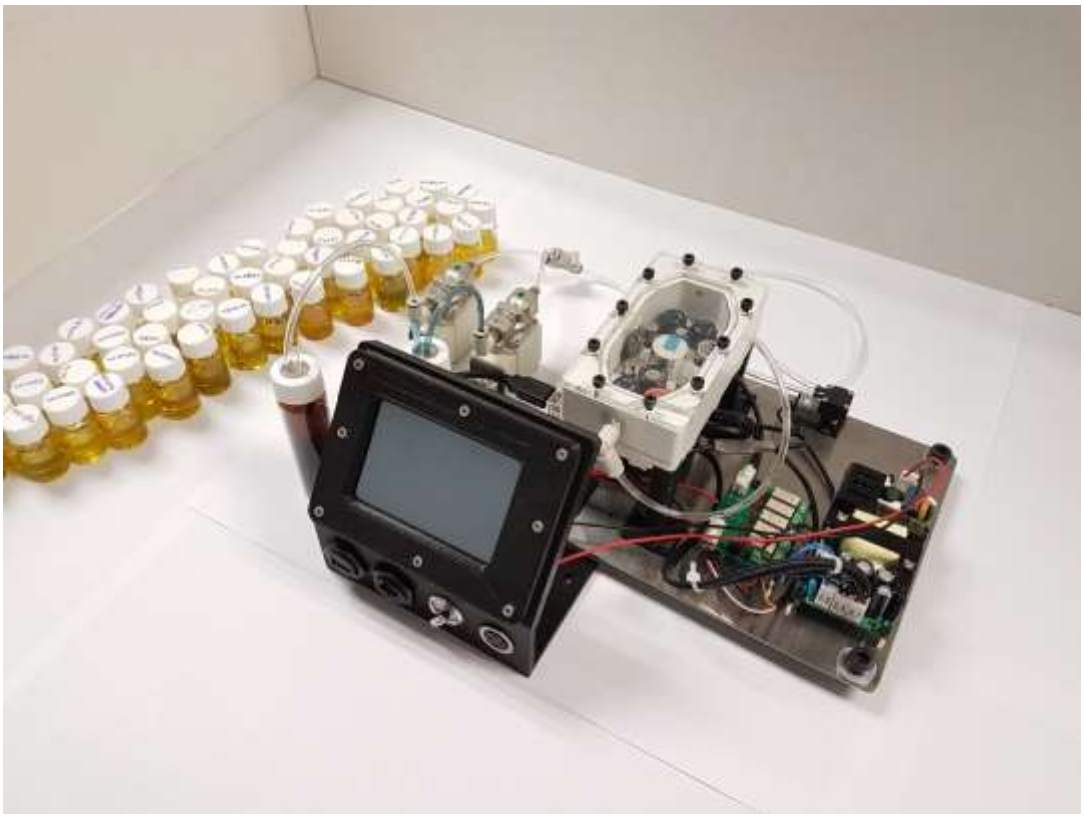


Figura 3.9 Prototipo 3

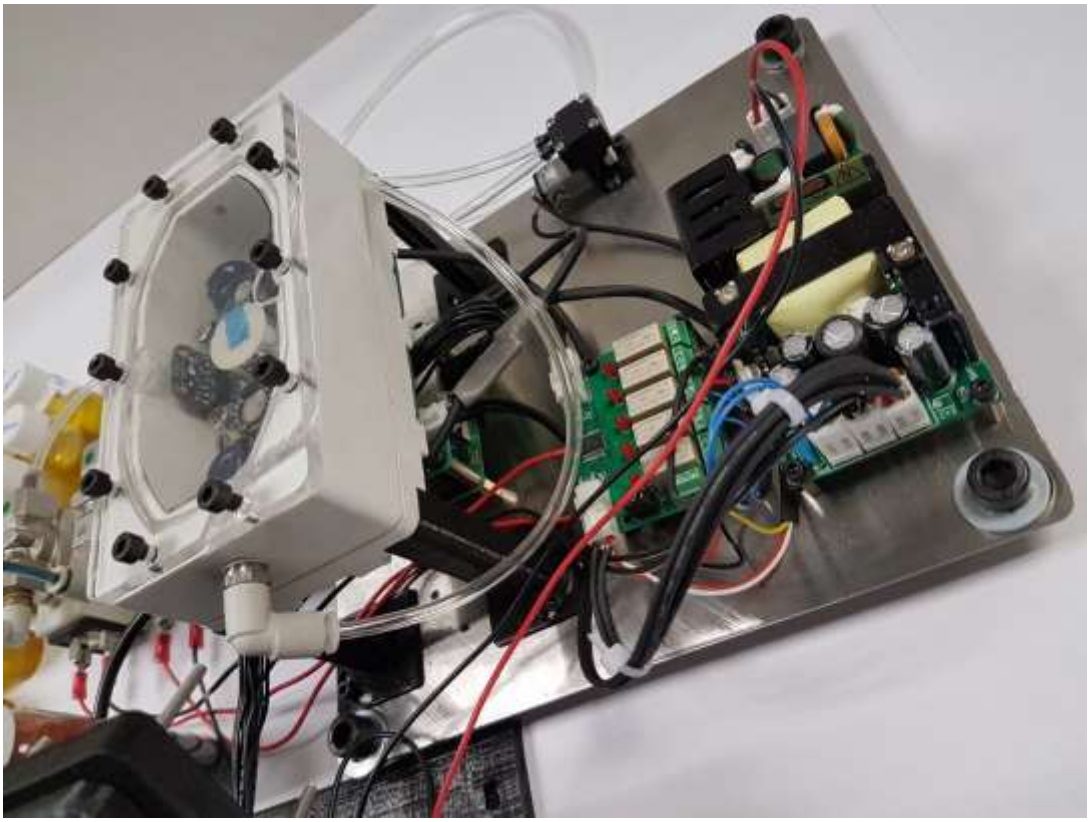


Figura 3.10 Prototipo 4

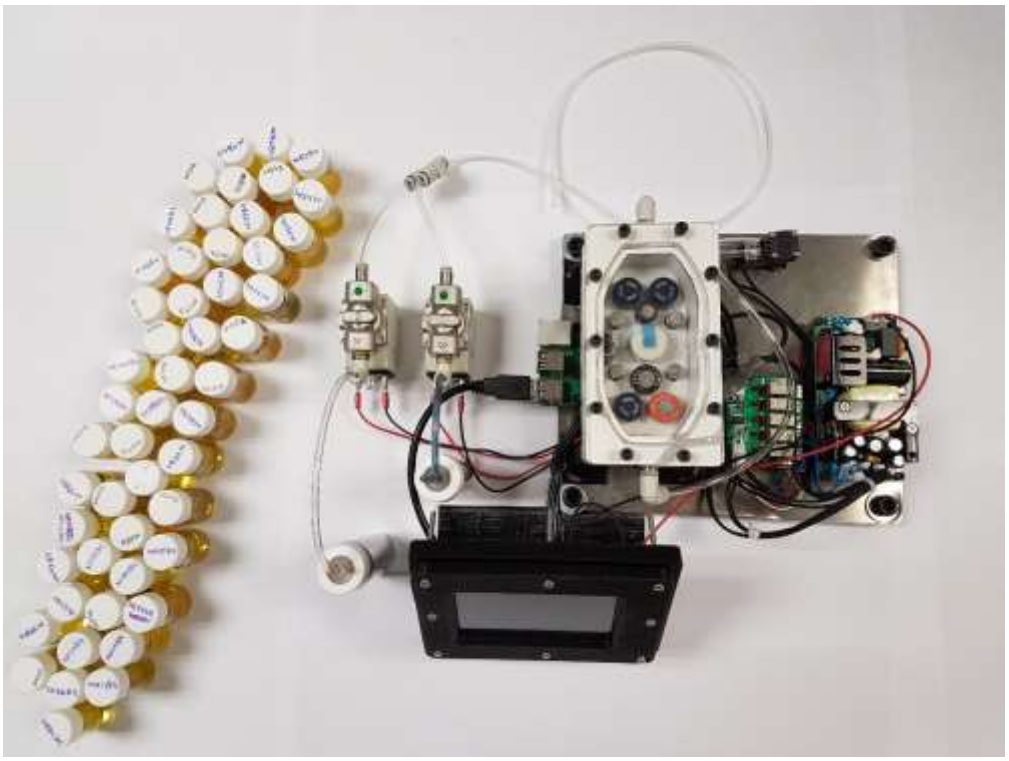


Figura 3.11 Prototipo 5

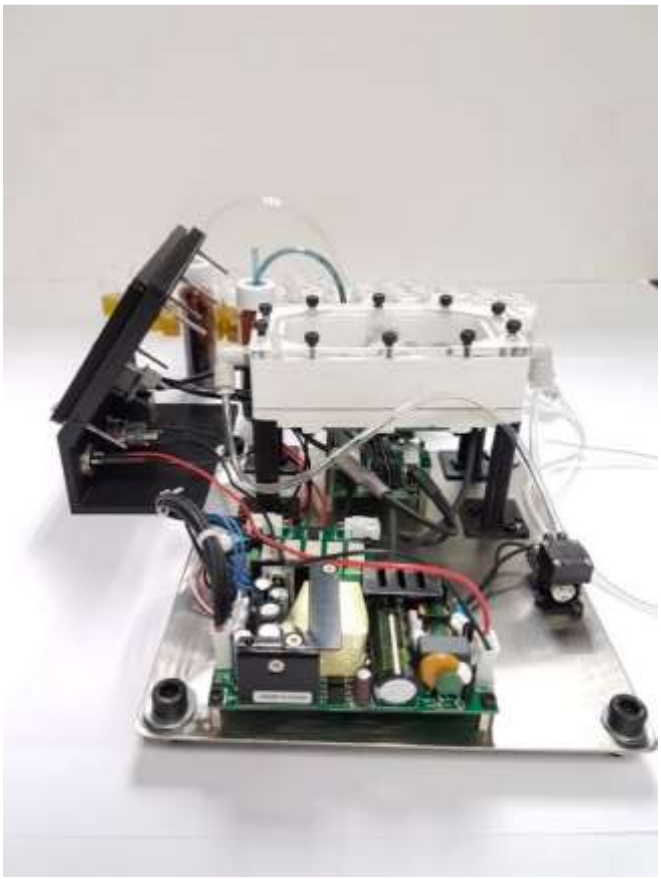


Figura 3.12 Prototipo 6

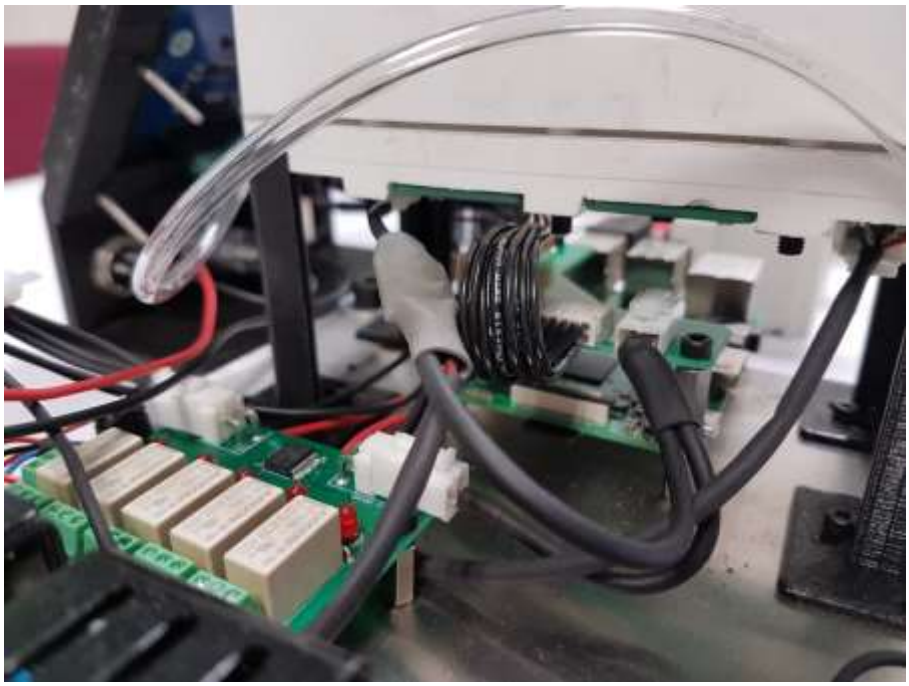


Figura 3.13 Prototipo 7

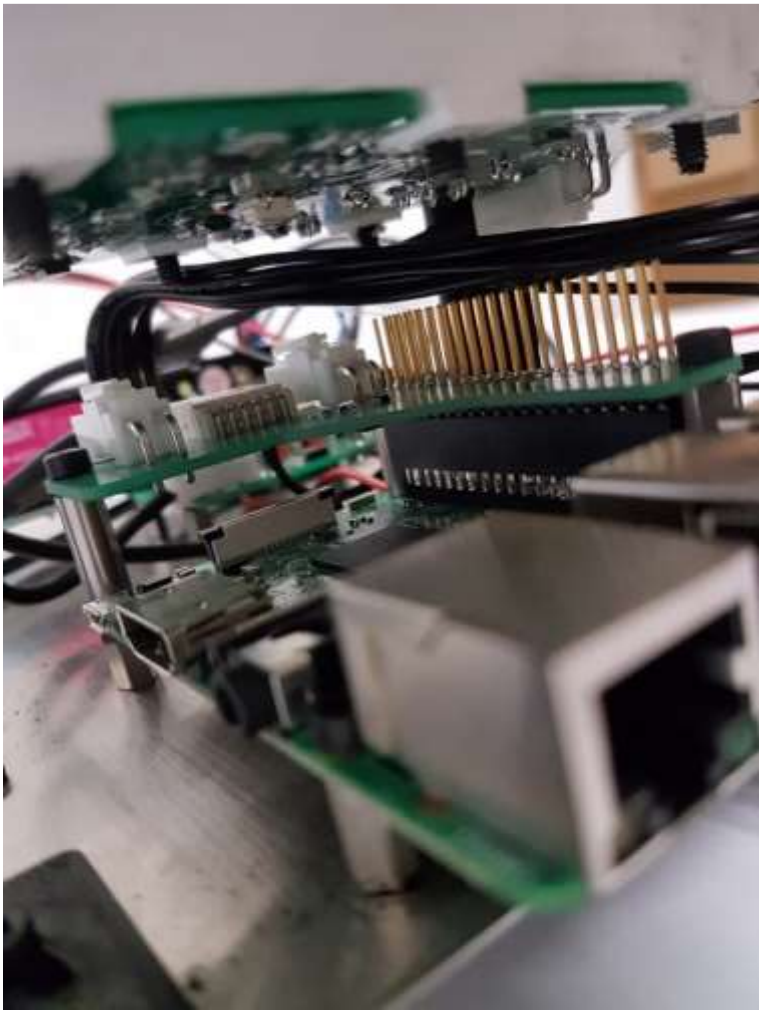


Figura 3.14 Detalle de Prototipo 8

Figura 3.15 Producto final 1

Figura 3.16 Producto final 2

3.3. Caso de estudio

Con el objetivo de hacer una primera evaluación de la nariz electrónica obtenida, se ha realizado un caso de estudio sencillo en el que se han pasado muestras de calidad catalogada mediante panel de cata y análisis químico con el fin de reconocer diferencias entre la calidad de las mismas. En principio solo se ha hecho diferencia entre aceite EVO, virgen extra, y aceite VO, aceite virgen.

MUESTRA	CATA	FRUTADO	AMARGO	PICANTE
M45	EVO	2.5	3.3	3.8
M46	EVO	2.7	3.3	3.2
M37	EVO	2.8	3.9	3.7
M51	EVO	2.8	3.3	3.3
M38	EVO	2.9	3.5	4
M48	EVO	3	3.2	3.5
M82	EVO	3	3.5	3.8
M81	EVO	3.1	3	3.3
M34	EVO	3.2	4.2	4.5
M27	EVO	3.3	3.9	3.6
M42	EVO	3.3	4.1	4.5
M35	EVO	3.4	3.5	3.4
M71	EVO	3.4	3.4	3.6
M21	EVO	3.5	3.7	3.5
M22	EVO	3.5	3.9	4.4
M24	EVO	3.5	3.6	4.1
M39	EVO	3.5	3.9	4
M66	EVO	3.5	3.6	3.8
M70	EVO	3.5	3.6	3.9
M83	EVO	3.5	3.3	3.5
M23	EVO	3.6	4	3.7
M32	EVO	3.6	3.5	3.5
M33	EVO	3.7	3.4	3.5
M36	EVO	3.7	3.2	3.2
M40	EVO	3.7	3.3	3.5
M50	EVO	3.7	3.2	3.3
M65	EVO	3.7	3.7	3.6
M31	EVO	3.8	3.8	3.5
M43	EVO	3.8	3.2	3.8
M44	EVO	3.8	3.8	3.3
M25	EVO	3.9	3.5	3.7
M26	EVO	3.9	3.6	3.6
M47	EVO	3.9	3.2	3.4
M30	EVO	4	3.8	4
M28	EVO	4.1	3.3	3.4
M41	EVO	4.2	3.6	3.5
M29	EVO	5.5	2.5	3

Tabla 3.1 Muestras de aceites EVO

La tabla superior corresponde a aceites virgen extra como se puede ver en la columna de la valoración del panel de cata. Las columnas del sabor del aceite son un indicador del grado de frutado, amargo o picante que tiene cada aceite.

MUESTRA	CATA	FRUTADO	AMARGO	PICANTE	defecto CM	MEDIA DEF
M53	VOO	2.2	3.8	3.5	avinado-avinagrado-agrio	2.3
M58	VOO	2.7	3.7	4.1	avinado-avinagrado-agrio	1.9
M61	VOO	2	3.3	3.6	avinado-avinagrado-agrio	3.3
M64	VOO	3.2	3.7	4	avinado-avinagrado-agrio	2.4
M67	VOO	1.5	3.2	3.3	avinado-avinagrado-agrio	3.4
M73	VOO	2.7	3	3.3	avinado-avinagrado-agrio	2.5
M77	VOO	2.6	3	3.5	avinado-avinagrado-agrio	2.5
M49	VOO	2.9	3.5	3.8	moho-humedad tierra	
M52	VOO	2.5	3.8	3.9	moho-humedad tierra	2.1
M54	VOO	3.2	3.5	3.2	moho-humedad tierra	1.8
M55	VOO	3.3	3.2	3.7	moho-humedad tierra	1.2
M56	LOO	1.2	3	2	moho-humedad tierra	4.2
M57	VOO	3.8	3.7	3.4	moho-humedad tierra	2
M59	VOO	3.2	3.5	3.8	moho-humedad tierra	2.1
M60	LOO	1.5	3.3	3.5	moho-humedad tierra	3.8
M62	VOO	3.5	3.2	3.2	moho-humedad tierra	1.7
M63	VOO	1.6	3.8	3.9	moho-humedad tierra	3.4
M68	VOO	3	3.2	3.1	moho-humedad tierra	1.5
M69	VOO	3.3	3.5	3.7	moho-humedad tierra	1.2
M72	VOO	1.5	3.1	3.8	moho-humedad tierra	3.3
M74	VOO	3	3.3	3.5	moho-humedad tierra	1.8
M75	VOO	2.5	2.6	3.5	moho-humedad tierra	1.6
M76	VOO	2.7	2.9	3.7	moho-humedad tierra	2.2
M78	VOO	2.8	2.4	3.1	moho-humedad tierra	2.3
M79	VOO	3	2.8	3	moho-humedad tierra	1.3
M80	VOO	2.2	2.6	3.3	moho-humedad tierra	3.3
M84	VOO	3.3	3.3	3	moho-humedad tierra	2.1

Tabla 3.2 Muestras de aceite VO

Para asegurar que los datos obtenidos de cada muestra fueran reales y comparables, se instauró un protocolo de medida de manera que siempre las muestras de aceite se midieran en las mismas condiciones:

- Encender la nariz electrónica y esperar a que la cámara de sensores alcance 50 ºC de temperatura aproximadamente. A partir de 50 ºC la temperatura siempre se estabiliza.
- Utilizar un filtro de carbono activo y sílice para limpiar la nariz electrónica en su proceso automático de limpieza.
- Calentar la muestra de aceite antes de ser medida hasta unos 40 ºC aproximadamente. Esto aumenta la concentración de componentes volátiles.

- Establecer los tiempos de medida y limpieza en 60 segundos respectivamente. De manera que el proceso completo dura 2 minutos.

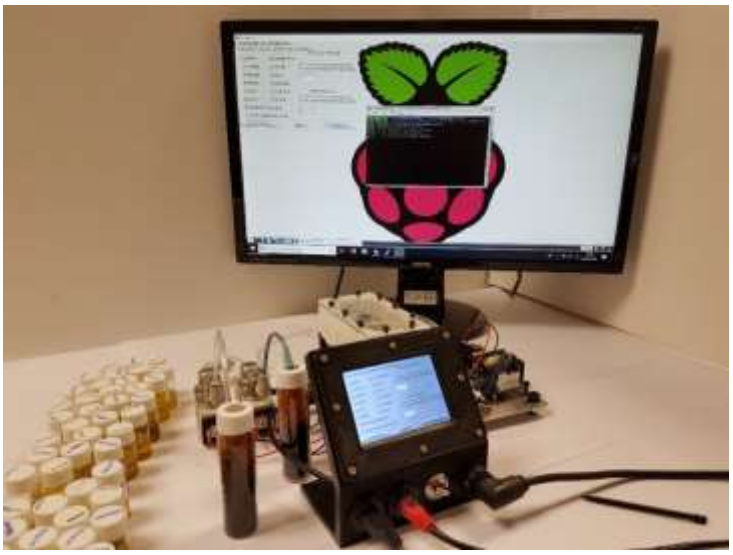


Figura 3.17 Montaje de medida 1

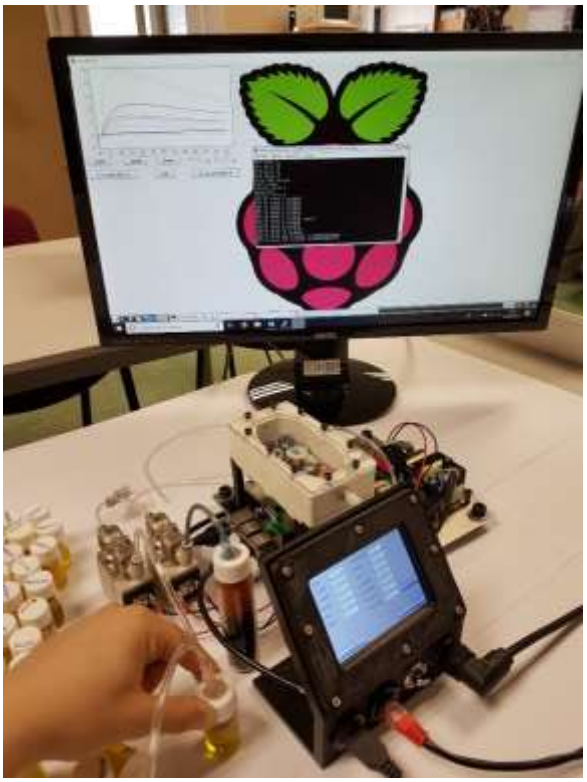


Figura 3.18 Montaje medida 2

A partir de la obtención de datos, se inicia el análisis de los mismos. En un primer análisis en bruto, se han obtenido las gráficas para cada muestra.

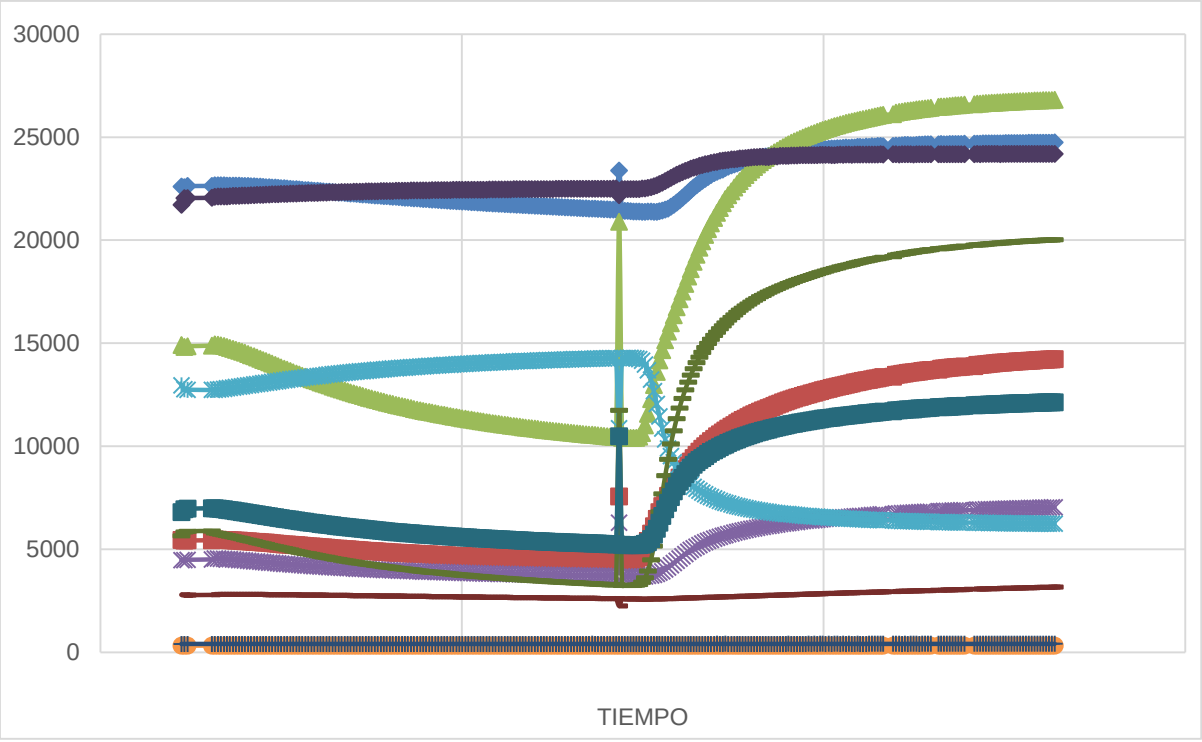


Figura 3.19 Datos absolutos de una muestra EVO en función del tiempo

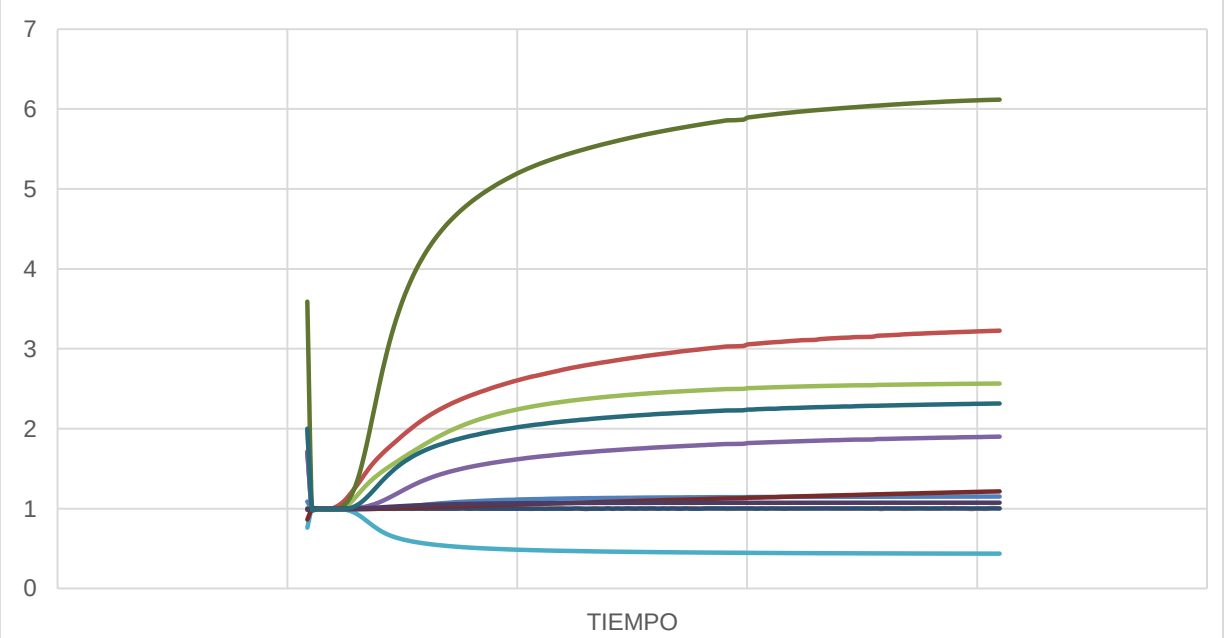


Figura 3.20 Datos referenciados de una muestra EVO en función del tiempo

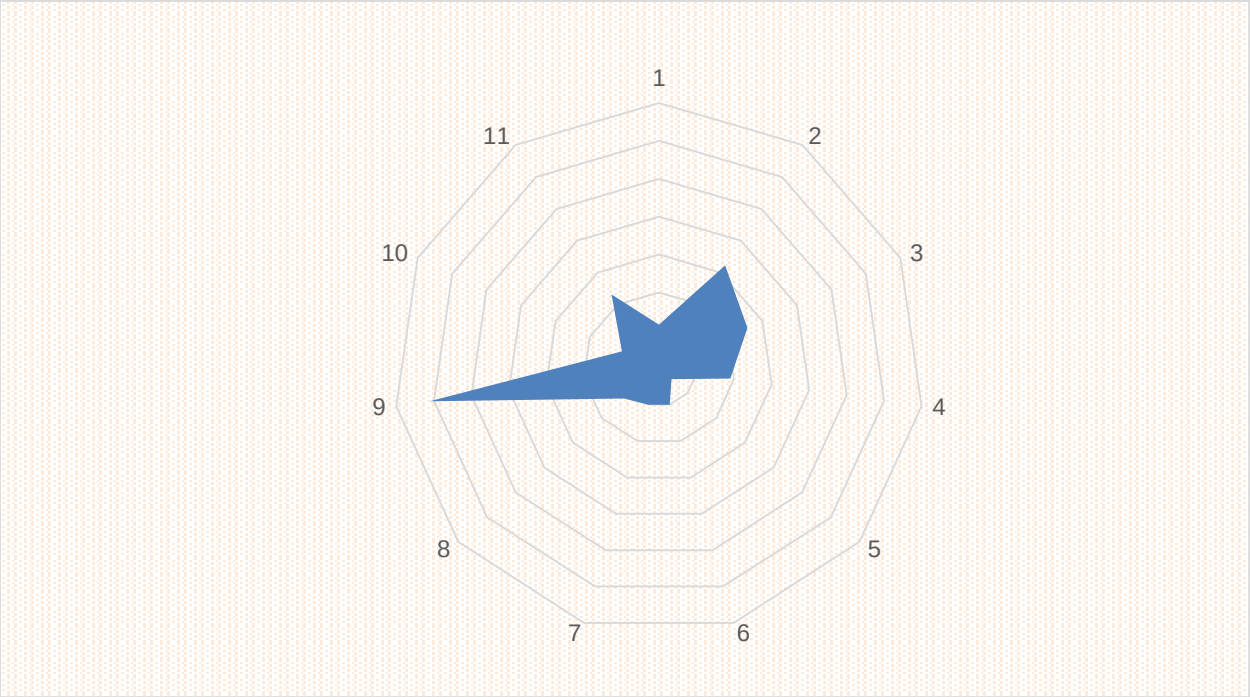


Figura 3.21 Huella de una muestra de aceite EVO

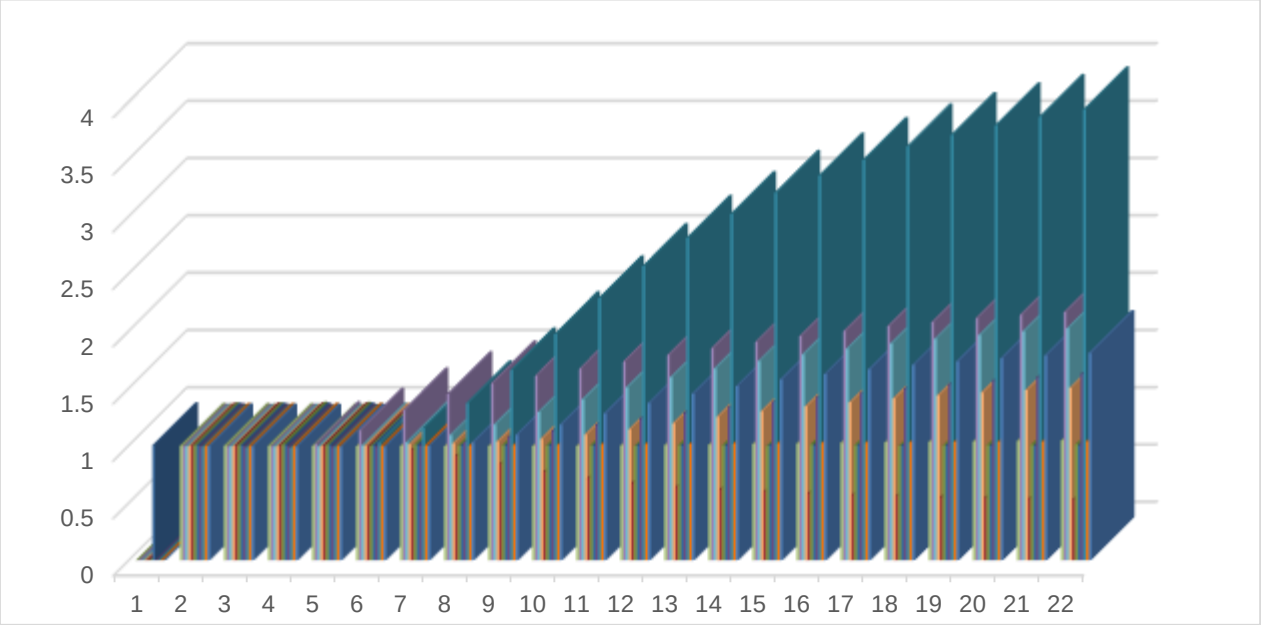


Figura 3.22 Nivel de excitación de los sensores en los primeros instantes de medida

Este procesado de la información es útil, sin embargo no ayuda a clasificar entre la calidad de un aceite a otro. Para este fin se han utilizado las siguientes técnicas de análisis avanzadas:

- Máquina soporte vectorial

Una Máquina de Soporte Vectorial (SVM) aprende la superficie decisión de dos clases distintas de los puntos de entrada. Como un clasificador de una sola clase, la descripción dada por los datos de los vectores de soporte es capaz de formar una frontera de decisión alrededor del dominio de los datos de aprendizaje con muy poco o ningún conocimiento de los datos fuera de esta frontera. (Betancourt, G.A., 2005)

- Regresión logística

Es un tipo de análisis de regresión utilizado para predecir el resultado de una variable en función de las variables predictores de entrada.

- Discriminante lineal

Es un método de clasificación supervisado de variables cualitativas en el que dos o más grupos son conocidos y nuevas observaciones se clasifican en uno de ellos en función de sus características.

El algoritmo que implementa estas técnicas lo aporta el Classification Learner Matlab App. Se basa en el concepto de Aprendizaje Automático o 'Machine Learning' en inglés. El Aprendizaje Automático lo podemos encontrar en reconocimiento facial, habla, publicidad sesgada, diagnósticos médicos, etc. Se trata de identificar las variables clave de un modelo, entrenarlo y evaluar los resultados del modelo.

Hay dos tipos de Aprendizaje Automático, como se puede ver en el siguiente esquema.

Ilustración 15 Machine Learning

Para el entrenamiento del Aprendizaje de Automático se han hecho dos tipos de validaciones de las muestras de datos, 'leave-one-out' y 'hold-out'. La primera consiste en dejar una muestra para validación cruzada y todas las demás para entrenamiento. Y la segunda, en dejar un porcentaje de entrenamiento y otro de validación. En este caso ha sido 60-40 respectivamente.

3.3.1. Preprocesado de la huella digital de las muestras de aceite virgen y virgen extra consideradas.

Los datos de las muestras catalogadas de aceite se han procesado posteriormente en tres matrices para su clasificación: Matriz de finales, de máximos y de promedios. Esto quiere decir, valor final, valor máximo y valor promedio de la medida.

Cada matriz tenía 11 columnas para los valores de los sensores y tantas filas como muestras se han medido. Además la matriz tenía una columna más con valores binarios para así poder etiquetar la calidad del aceite. Con un 1, se etiquetaba el aceite virgen extra. Con un 0, el aceite virgen.

Estas matrices se hacen analizar en el 'Classification Learner' de Matlab y los resultados que se obtienen corresponden a la clasificación que se mostrará en el siguiente punto 3.3.2.

NUBE DE PUNTOS: MATRIZ DE MÁXIMOS

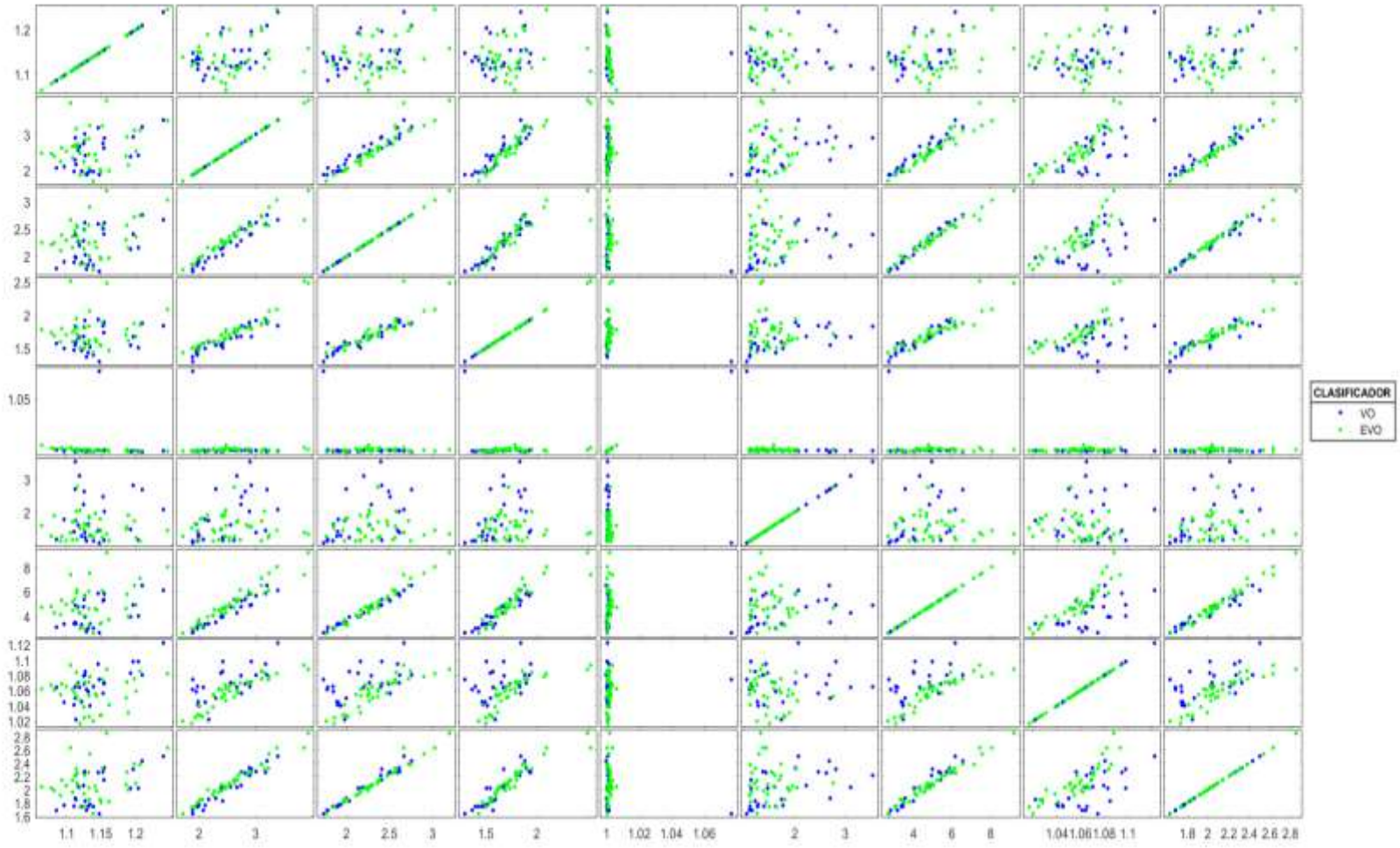


Figura 3.23 Nube de puntos. Matriz de máximos

NUBE DE PUNTOS: MATRIZ DE PROMEDIOS

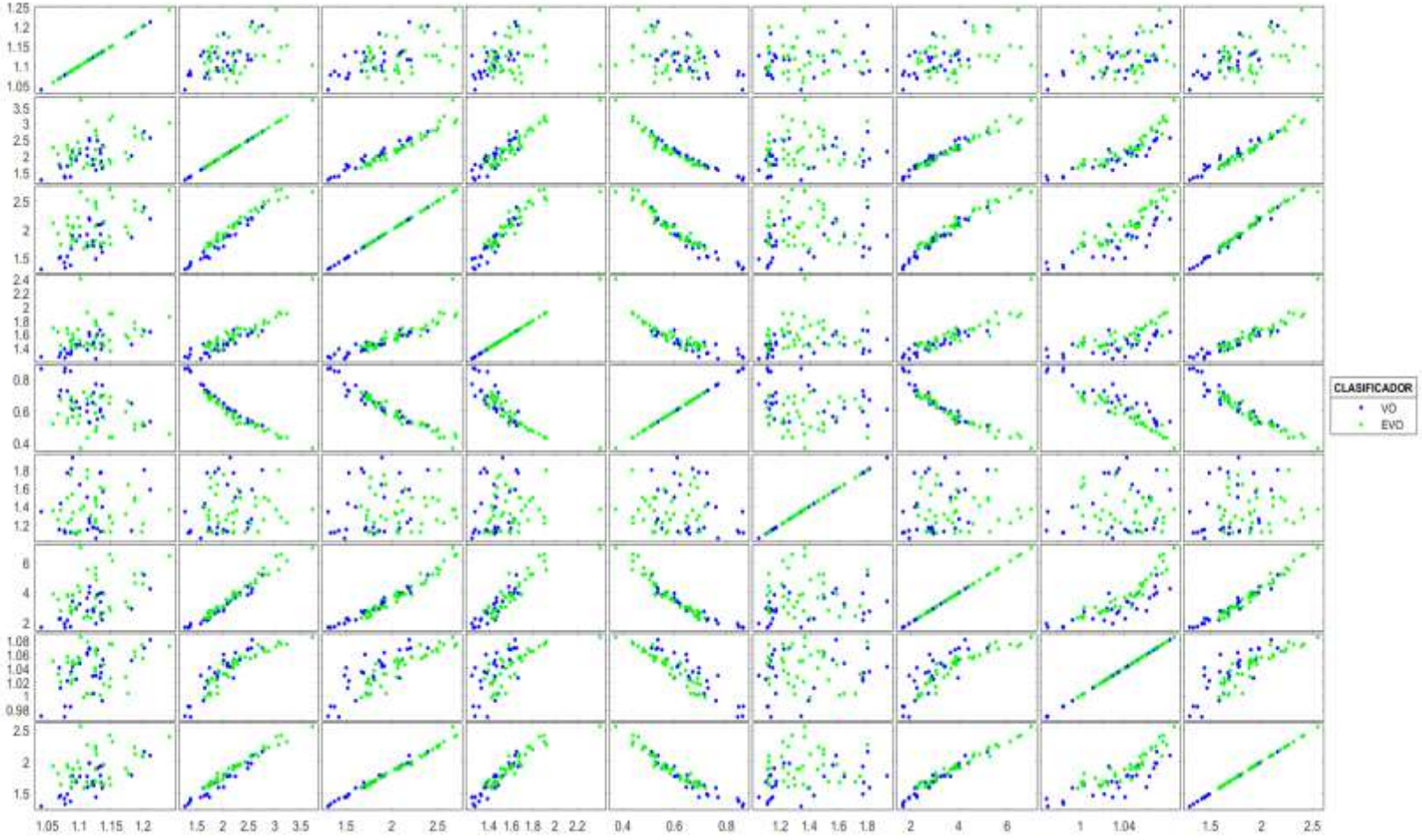


Figura 3.24 Nube de puntos. Matriz de promedios

NUBE DE PUNTOS: MATRIZ DE FINALES

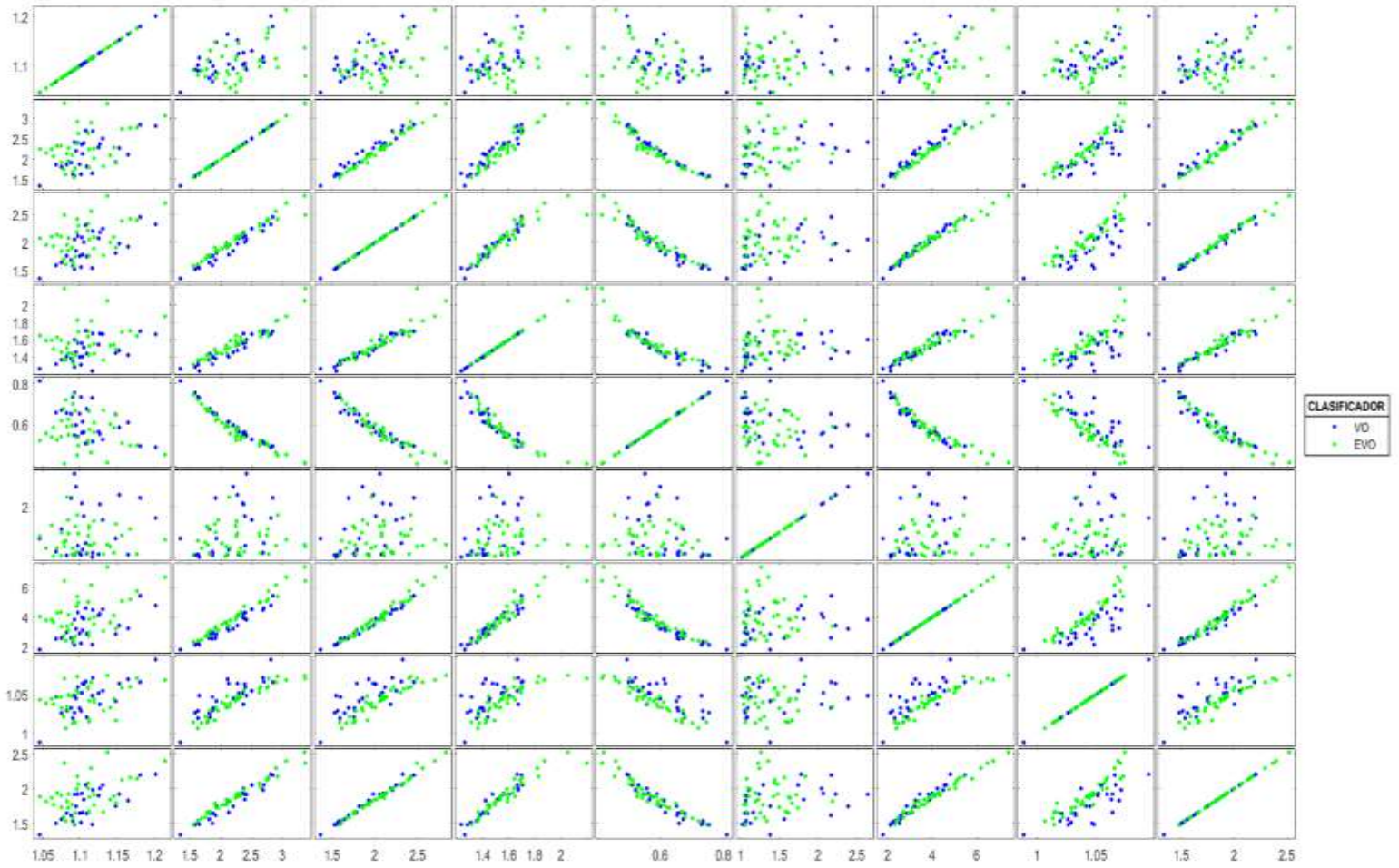


Figura 3.25 Nube de puntos. Matriz de finales

Las figuras 3.23, 3.24 y 3.25, muestran una nube de puntos que representan las dos clases analizadas en el caso de estudio. Los puntos azules son las muestras VO y los puntos verdes son las muestras EVO. Hay 9x9 gráficas representadas, que corresponden a 9 sensores de la nariz electrónica que aportan información diferenciadora entre clases.

La matriz diagonal de gráficas es lineal porque los puntos de cada muestran corresponde al mismo sensor, es decir, sensor 1 frente a sensor 1 en la primera columna y primera fila. Sensor 2 frente a sensor 2 en la segunda columna y segunda fila. Y así sucesivamente.

3.3.2. Resultados de la clasificación EVO / VO

Los resultados se muestran en formato de matriz de confusión. La matriz de confusión es una herramienta de visualización que se emplea en el aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real.

Matriz datos finales: Validación cruzada ‘leave-one-out’.

- Mejor resultado con clasificador máquina soporte vectorial: **79,7 % de acierto**.

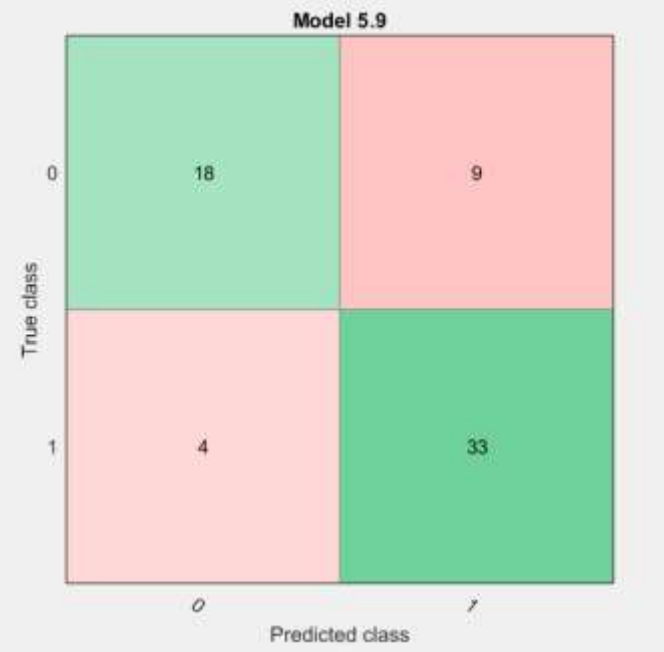


Figura 3.26 Matriz de confusión 1

Matriz datos finales: Validación ‘hold-out’ 60-40.

- Mejor resultado con clasificador máquina soporte vectorial: **84 % de acierto**.

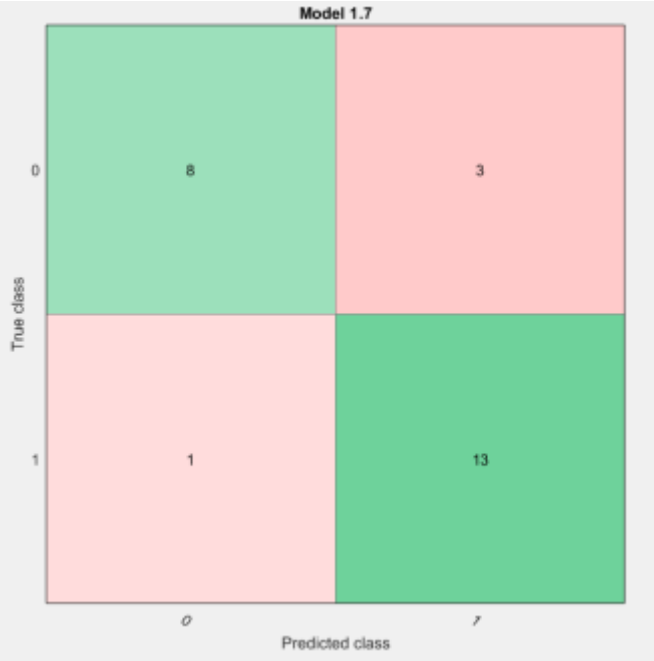


Figura 3.27 Matriz de confusión 2

Matriz datos máximos: Validación cruzada ‘leave-one-out’.

- Mejor resultado con clasificador máquina soporte vectorial: **81,3 % de acierto**.

Figura 3.28 Matriz de confusión 3

Matriz datos máximos: Validación ‘hold-out’ 60-40.

- Mejor resultado con regresión logística: **92 % de acierto**.

Figura 3.29 Matriz de confusión 4

Matriz datos promedios: Validación cruzada ‘leave-one-out’.

- Mejor resultado con discriminante lineal: **81,3 % de acierto**.

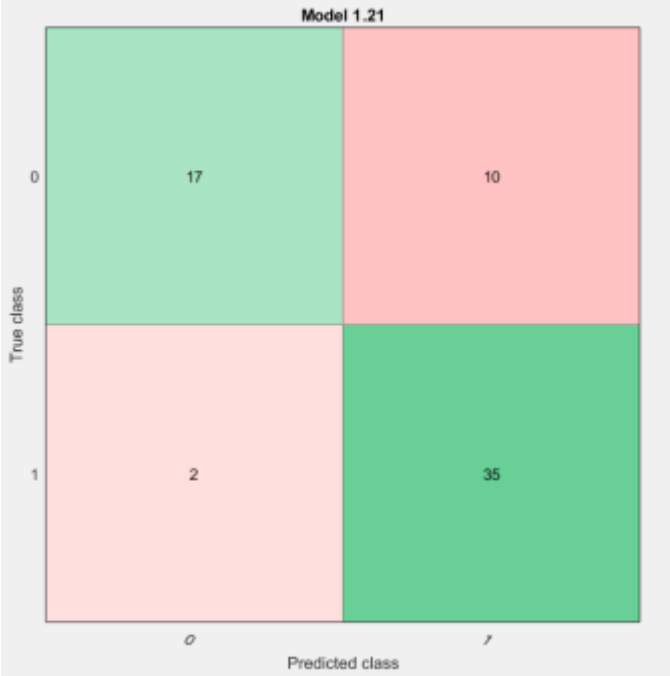


Figura 3.30 Matriz de confusión 5

Matriz datos promedios: Validación ‘hold-out’ 60-40.

- Mejor resultado con discriminante lineal: **84 % de acierto**.

Figura 3.31 Matriz de confusión 6

Los resultados que arrojan las matrices de confusión son realmente buenos tratándose de la primera versión de la nariz electrónica. Como veremos en el capítulo 4, el rango de mejora es aún muy amplio y el camino por recorrer es largo. Por ello, obtener resultados de acierto altos en la primera validación es realmente un éxito.

Destaca el 92 % que se obtiene en la matriz de máximos aplicando la técnica de regresión logística con validación ‘hold-out’. Aun siendo una distinción simple entre dos clases de aceite, EVO y VO, predice la fiabilidad que puede llegar a tener la nariz electrónica en el análisis de las propiedades organolépticas de un aceite.

4. CONCLUSIONES Y TRABAJOS FUTUROS

La nariz electrónica ha resultado un proyecto muy completo, obteniendo un producto cuya evolución esperada a corto plazo, aun siendo un prototipo, es la de acabar teniendo entidad y garantía suficiente como para competir en el mercado.

Las tres partes del proyecto se han acabado satisfactoriamente, cumpliendo los plazos fijados. De hecho antes de la entrega y presentación de los trabajos ya se ha empezado a idear y diseñar las nuevas versiones de la nariz electrónica con el fin de adelantar plazos y que una versión mejorada de la nariz puede estar operativa en varias almazaras de la provincia para la campaña 2018/2019.

Los objetivos que se habían marcado en este trabajo también se han cumplido, desde el aprendizaje de las herramientas software, el diseño de la aplicación haciendo uso de ellas, implementar dicha aplicación y la validación de la aplicación desarrollada con el caso de estudio explicado en el punto anterior. De este último objetivo destaca el dato más favorable de los resultados del punto 3.3.2., un 92% de acierto discriminando entre aceite virgen y virgen extra.

Es muy gratificante realizar un proyecto colaborativo y conseguir que salga adelante con la certeza de que no solo es un trabajo de fin de grado, sino que también puede ser útil en la industria. Pensamos que hemos colaborado así con la investigación y el desarrollo de mejoras en los procesos de obtención de AOVE además de ayudar a objetivar los parámetros de calidad del aceite. Desde un punto de vista académico, la participación en este proyecto ha aportado también al autor de esta parte del trabajo conocimientos los cuales no había abordado durante los estudios del grado: el dominio de la programación en Java, el uso avanzado de Raspberry y la realización de un proyecto real y de cierta envergadura desde cero hasta el final.

Volviendo a la línea de los avances esperables en un nuevo modelo de nariz electrónica, creemos que todas las partes del proyecto tienen un largo recorrido de mejora. Solo vamos a destacar el futuro del software, que tiene mucho que ver con la nueva revolución industrial, conocida como la Industria 4.0.

4.1. Nariz Electrónica online

Podríamos hablar de la primera versión de la nariz electrónica como una nariz offline. Bien es cierto que se ha tratado de incluir el IoT con la opción de enviar correos electrónicos con los resultados de la medida, pero esto sería solo el comienzo. La nariz offline quiere decir que no hay conexión inalámbrica entre este dispositivo y otros dispositivos, como móviles, tabletas u ordenadores portátiles. Esta conexión podría conseguirse mediante varias tecnologías, entre las cuales, sin duda, la más destacada sería la conexión a Internet. Otras tecnologías posibles para conectar dispositivos son por ejemplo Bluetooth o GSM.

La finalidad de dar este paso es que la nariz deje de ser una máquina con funcionalidad exclusiva in situ, sino que también podría utilizarse y trabajar con ella en otros ámbitos o en otros lugares, no necesariamente en una almazara.

Podríamos tener los datos instantáneos de la calidad del aceite en nuestro propio dispositivo móvil mediante una app para móviles, Android o iOS. Incluso la configuración de la medida de la nariz electrónica que hemos visto, que actualmente se hace desde la pantalla táctil, podríamos hacerla desde nuestro propio dispositivo.

Las posibilidades de conectividad que ofrece la Raspberry Pi 3 B empleada y las posibilidades de futuros modelos, hacen que el IoT en la nariz electrónica sea un objetivo casi obligatorio.

Ilustración 16 Futuros pasos de la nariz electrónica

Tras la conectividad de la nariz electrónica, el siguiente paso sería pasar al concepto de nariz electrónica online. La diferencia entre ambos es sutil pero existe y da un sentido más amplio a la transversalidad en el uso de la nariz. La idea es que todo se gestione desde la nube, con una gran base de datos que coleccione todos los datos, las gráficas, resultados y muchos más elementos que aún están por diseñar.

Este paso daría a la nariz electrónica un matiz fundamental para su comercialización y su garantía de calidad. Dentro de los proyectos del sector Agro

que se realizan en GRAV, la nariz podría integrarse en el servicio informático ofrecido. El abanico de posibilidades es grande, la dirección de la nariz electrónica es sin duda la Industria 4.0, más aún de lo que es ya.

4.2. Herramientas de análisis de datos: Big Data

El Big Data parece un concepto muy alejado de la realidad de este proyecto pero en cierto modo es esencial. Como se ha comentado en algún momento durante el trabajo, la capacidad de la matriz de sensores de la nariz es solamente medir, extraer datos de cada medida. Los datos extraídos por los sensores realmente no nos dicen nada considerándolos aislados uno a uno. La respuesta de la nariz electrónica debe considerarse después de analizar, cruzar y clasificar todos los datos de muchas muestras de aceite de entrenamiento. En términos comprensibles, estaríamos hablando de entrenar a la nariz electrónica, prepararla para reconocer o discriminar un aceite según su calidad.

Existen muchas técnicas para ello, todas bajo el prisma del Big Data y el concepto que hay detrás de él. En el capítulo de resultados hemos visto como se han aplicado clasificadores para las muestras de aceite catalogadas que ya se habían pasado por la nariz con el fin de entrenarla. Sin embargo, un entrenamiento mejor y más avanzado es posible si se utilizan técnicas avanzadas de análisis de datos. Esta es una meta deseable para la nariz a medio plazo, donde no solo se crucen datos de una misma almazara o tipo de aceituna, sino que pueda ser capaz de distinguir entre distintas variedades de cualquier tipo y ajustar al máximo los parámetros de calidad.

Para ello, otro factor importante es el uso de los sensores adecuados. No queremos decir que los sensores utilizados no sean los adecuados, sino que existen tecnologías más avanzadas y precisas. Incluso con la misma tecnología, hay sensores que miden otros volátiles que pudieran resultar más interesantes para nuestra aplicación. Sin duda el rango de mejora es amplío, aunque el producto al que se ha llegado sea ya bueno y asequible para el mercado. Si avanzáramos con todas estas mejoras, el precio de la nariz lógicamente también subiría y habría que ver hasta qué punto podríamos elevar el coste de producción manteniendo la competitividad. Da para otro trabajo sobre estudio de mercado en innovación e investigación en la producción de AOVE en Jaén.

5. PRESUPUESTO

PRESUPUESTO, DESCOMPUESTOS Y MEDICIONES

CÓDIGO	RESUMEN			CANTIDAD	PRECIO	IMPORTE
1	**ELECTRÓNICA**					
A	**u HARDWARE**					
	Descomposición:					
	896-8660	*u*	*Raspberry Pi3B*	*1.00*	*29.470*	*29.470*
	124-9640	*u*	*MicroSD 16GB Raspberry*	*1.00*	*11.200*	*11.200*
	706-6212	*u*	*TRACOPOWER TOP 60252*	*1.00*	*42.170*	*42.170*
	709-4544	*u*	*Convertidor A/D 16bits ADS1115*	*3.00*	*5.310*	*15.930*
	504-5086	*u*	*Regulador LDO 10V 500mA*	*1.00*	*1.470*	*1.470*
	770-9928	*u*	*Regulador LDO 3V 800mA*	*1.00*	*0.713*	*0.713*
	2492376	*u*	*Regulador LDO 0.9V 300mA*	*1.00*	*0.379*	*0.379*
	2328474	*u*	*Potenciometro lineal 100Ohm SMD*	*1.00*	*2.200*	*2.200*
	901-3737	*u*	*Resistencia SMD 47kOhm 1206*	*1.00*	*0.008*	*0.008*
	901-3733	*u*	*Resistencia SMD 10kOhm 1206*	*29.00*	*0.008*	*0.232*
	223-2344	*u*	*Resistencia SMD 3.9kOhm 1206*	*2.00*	*0.008*	*0.016*

PRESUPUESTO, DESCOMPUESTOS Y MEDICIONES

CÓDIGO	RESUMEN			CANTIDAD	PRECIO	IMPORTE
	812-6973	*u*	*Resistencia SMD 1.2kOhm 1206*	*2.00*	*0.027*	*0.054*
	679-2225	*u*	*Resistencia SMD 560Ohm*	*5.00*	*0.024*	*0.120*
	766-1126	*u*	*Condensador SMD 100nF 1206*	*3.00*	*0.014*	*0.042*
	766-1141	*u*	*Condensador SMD 1uF 1206*	*6.00*	*0.039*	*0.234*
	173-4070	*u*	*Condensador SMD 10uF 1210*	*3.00*	*0.118*	*0.354*
	906-0294	*u*	*Conector Macho PCB 90º 3 pines*	*1.00*	*0.905*	*0.905*
	820-1175	*u*	*Carcasa conector JST 3 pines*	*1.00*	*0.494*	*0.494*
	9492038	*u*	*Conector Macho PCB 90º 2 pines*	*5.00*	*0.120*	*0.600*
	630470	*u*	*Carcasa conector JST 2 pines*	*5.00*	*0.038*	*0.190*
	254-6110	*u*	*Conector hembra PCB 2x20 2.54mm*	*1.00*	*7.020*	*7.020*
	825-3514	*u*	*Level Shifter I2C TCA9517*	*1.00*	*0.876*	*0.876*
	509-706	*u*	*Transistor Pair BCM857BS*	*1.00*	*0.131*	*0.131*
	827-0522	*u*	*Transistor Mosfet DMG2305UX Canal P*	*1.00*	*0.084*	*0.084*
	2399203	*u*	*Conector Macho PCB 90º 2x6*	*1.00*	*0.392*	*0.392*
	1830789	*u*	*Conector Macho PCB 2x6*	*1.00*	*0.385*	*0.385*

PRESUPUESTO, DESCOMPUESTOS Y MEDICIONES

CÓDIGO	RESUMEN			CANTIDAD	PRECIO	IMPORTE
	1830784	*u*	*Carcasa conector JST 2x6*	*2.00*	*0.182*	*0.364*
	1830725	*u*	*Terminal crimpado JST serie PHD*	*24.00*	*0.026*	*0.624*
	770-9707	*u*	*Expansor E/S MCP23008*	*1.00*	*1.104*	*1.104*
	256-0649	*u*	*Relé DPDT PCB 5V*	*5.00*	*2.520*	*12.600*
	725-9344	*u*	*Transistor Mosfet canal N 5.3ª*	*5.00*	*0.277*	*1.385*
	220-4276	*u*	*Bloque de terminales PCB 3 vías*	*6.00*	*0.832*	*4.992*
	1301674	*u*	*Perla de Ferrita 0805 1.5kOhm*	*4.00*	*0.066*	*0.264*
	842-8009	*u*	*Diodo Rectificador 1N4001*	*5.00*	*0.018*	*0.090*
	168-9443	*u*	*Diodo LED Rojo orificio pasante*	*5.00*	*0.048*	*0.240*
	863-2844	*u*	*Conector Macho 2x3 PCB jumper*	*1.00*	*0.144*	*0.144*
	863-2762	*u*	*Jumper 2.54mm*	*3.00*	*0.093*	*0.279*
	2473872	*u*	*Raspberry 7" touchscreen*	*1.00*	*50.190*	*50.190*
	Total cantidades alzadas			*1.00*		
				1.00	*187.945*	*187.945*

B **u SOCKETS**

Descomposición:

PRESUPUESTO, DESCOMPUESTOS Y MEDICIONES

CÓDIGO	RESUMEN			CANTIDAD	PRECIO	IMPORTE
	19683	*u*	*R375-0805-01T-433E-R27-L14*	*3.00*	*3.980*	*11.940*
	19960	*u*	*R200-SP03-01T-295V-R27-L14*	*2.00*	*3.590*	*7.180*
	20030	*u*	*R200-0404-01T-01S-R27-L14*	*2.00*	*2.170*	*4.340*
	20370	*u*	*R530-0403-02N-436P55-R27-L14*	*1.00*	*4.880*	*4.880*
	20683	*u*	*R375-0806-01T-433E-R27-L14*	*2.00*	*4.190*	*8.380*
	22667	*u*	*R240-SP04-01T-295V-R27-L14*	*1.00*	*4.790*	*4.790*
	Total cantidades alzadas			*1.00*		
				1.00	*41.510*	*41.510*
C	**u SENSORES**					
	Descomposición:					
	TGS2620-C00	*u*	*GAS SENSOR SOLVENT VAPOUR*	*1.00*	*9.900*	*9.900*
	TGS2602-B00	*u*	*SENSOR VOC*	*1.00*	*6.900*	*6.900*
	SK-25F	*u*	*SENSOR OXIGENO*	*1.00*	*28.800*	*28.800*
	TGS822	*u*	*GAS SENSOR SOLVENT VAPOUR ALCOHOL*	*1.00*	*6.700*	*6.700*
	TGS832-A00	*u*	*SENSOR GASES REFRIG*	*1.00*	*13.600*	*13.600*
	TGS6812-D00	*u*	*SENSOR CATALITICO H2*	*1.00*	*15.750*	*15.750*
	SP-31-00	*u*	*PROPANO Y BUTANO*	*1.00*	*35.650*	*35.650*

PRESUPUESTO, DESCOMPUESTOS Y MEDICIONES

CÓDIGO	RESUMEN			CANTIDAD	PRECIO	IMPORTE
	SP3S-AQ2-01	*u*	*AIR QUALITY CONTROL/ AUTOMOTIVE*	*1.00*	*23.850*	*23.850*
	SP-11-00	*u*	*HYDROCARBONS GAS DETECTOR*	*1.00*	*35.650*	*35.650*
	SP-19-01	*u*	*HYDROGEN DETECTION*	*1.00*	*35.650*	*35.650*
	SB-51-00	*u*	*SH2 DETECTOR*	*1.00*	*35.650*	*35.650*
	Total cantidades alzadas			*1.00*		
				1.00	*248.100*	*248.100*
D	**u**	**INTERFAZ EXTERIOR**				
	Descomposición:					
	111-6761	*u*	*Zócalo USB Panel*	*1.00*	*15.760*	*15.760*
	111-6747	*u*	*Zócalo RJ45 Apantallado*	*1.00*	*16.120*	*16.120*
	125-7926	*u*	*Tapón de plástico IP67*	*2.00*	*4.570*	*9.140*
	261-6180	*u*	*Conector Hembra 3 pines Binder*	*1.00*	*5.250*	*5.250*
	734-5874	*u*	*Conector Aéreo Acodado 3 pines Binder*	*1.00*	*20.750*	*20.750*
	877-1974	*u*	*Interruptor Giratorio IP67*	*1.00*	*13.870*	*13.870*
	707-9450	*u*	*Conector Hembra 6 pines panel Lumberg*	*2.00*	*8.610*	*17.220*
	909-2250	*u*	*Conector Aéreo Recto 6 pines Lumberg*	*2.00*	*15.620*	*31.240*

PRESUPUESTO, DESCOMPUESTOS Y MEDICIONES

CÓDIGO	RESUMEN	CANTIDAD	PRECIO	IMPORTE
	Total cantidades alzadas	*1.00*		
		1.00	*129.350*	*129.350*

TOTAL 1..... 606.905

PRESUPUESTO, DESCOMPUESTOS Y MEDICIONES

CÓDIGO	RESUMEN	CANTIDAD	PRECIO	IMPORTE
2	**MECÁNICA**			
22046403046567	**u BOMBA KNF DC 5V NMP05KPS**	*1.00*		
		1.00	*71.460*	*71.460*
	Total cantidades alzadas			
0201567425002	**u RACOR KFG2H0425-00 #**			
	Racor para conexiones E/S sobre panel exterior - Modelo SMC KGE04-00	*6.00*		
	Total cantidades alzadas	*6.00*	*13.780*	*82.680*
0201562425106	**u RACOR KFG2L0425-01S #**			
	Racor acodado para salidas de la cámara de sensores - Modelo SMC KGL04-01S	*2.00*		
	Total cantidades alzadas	*2.00*	*12.110*	*24.220*
0205704254203	**u TUBO TH0425N-20 ROLLO #**	*1.00*		
	Total cantidades alzadas	*1.00*	*48.440*	*48.440*
0203311216553	**u ELECTROVÁLVULA VQD1121-6LO-M5-Q**	*2.00*		
	Total cantidades alzadas	*2.00*	*24.300*	*48.600*
0203051661149	**U CONECTOR AXT661-14A-10 #**	*2.00*		
	Total cantidades alzadas	*2.00*	*2.000*	*4.000*
8AC-2500	**u CAJA ALUCASE AC 252 DA 191.250.100**			
	Caja metálica IP67 - Oferta MALLOL ASETYC O18-02466-P3B7X8	*1.00*		
	Total cantidades alzadas	*1.00*	*152.100*	*152.100*
0101180001	**u MECANIZADO DE PLACA MOVIL**			
	Mecanizado del cuerpo de la caja metálica para situación de E/S de la Nariz Electrónica. - Oferta CEMTEC ENGINEERING 01-02-180002			
	Total cantidades alzadas	*1.00*		
0102180002	**u MODIFICACIÓN DE CAJA PROTECTORA**	*1.00*	*46.300*	*46.300*
	Mecanizado de la tapadera exterior de la caja metálica para colocación de la pantalla. . - Oferta CEMTEC ENGINEERING 01-01-180001	*1.00*		
	Total cantidades alzadas	*1.00*	*29.300*	*29.300*
00094450	**u TORNILLO M4x50 ACERO INOX.**	*10.00*		
	Total cantidades alzadas	*10.00*	*0.220*	*2.200*
AG00001	**kg MATERIAL PLÁSTICO ARNITE**	*1.43*		

	Total cantidades alzadas	*1.43*	*10.500*	*15.015*
PC00001	**u PLANCHA POLICARBONATO 148x210mm**	*1.00*		
	Total cantidades alzadas	*1.00*	*6.770*	*6.770*

TOTAL 2.....531.085

RESUMEN DE PRESUPUESTO MATERIAL

CAPÍTULO	RESUMEN	IMPORTE	%
1	ELECTRÓNICA.....	606.905	53.33
2	MECÁNICA.....	531.085	46.67

PRESUPUESTO DE EJECUCIÓN MATERIAL	**1,137.990**
21% IVA.....	238.978
PRESUPUESTO BASE DE LICITACIÓN	**1,376.968**

Asciende el presupuesto a la expresada cantidad de MIL TRESCIENTOS SETENTA Y SEIS EUROS con NOVENTA Y SEIS CÉNTIMOS

7 de junio 2018.

6. BIBLIOGRAFÍA Y REFERENCIAS

Bibliografía y referencias por orden alfabético

Adafruit 4-Channel ADC. https://learn.adafruit.com/adafruit-4-channel-adc-breakouts/downloads

Beltrán Ortega, J.; Gámez García, J.; and Gómez Ortega, J. *Precision of volatile compound analysis in extra virgin olive oil: The influence of MOS electronic nose acquisition factors*. 2015, IEEE International Conference on Industrial Technology (ICIT)

Betancourt, G.A. (2005). *LAS MÁQUINAS DE SOPORTE VECTORIAL (SVMs)*

Boeker, P. (Julio 2014). *On 'Electronic Nose' methodology.* Sensors and Actuators B: Chemical, julio 2014, Vol.204, pág. 2-17.

Frambuesa Pi Colombia. http://frambuesapi.co/

Fundamentos del protocolo I2C. http://learn.teslabem.com/fundamentos-del-protocolo-i2c-aprende/

Gemelli, M. (2017). *Smart Sensors Fulfilling The Promise Of The IoT*

Gómez García, J. (2018). *Desarrollo mecánico de una nariz electrónica para la discriminación de aceites.* Trabajo Fin de Grado.

Hernández Cledera, M. (2018). *Desarrollo hardware de una nariz electrónica para la discriminación de aceites.* Trabajo Fin de Grado.

I2C. https://es.wikipedia.org/wiki/I%C2%B2C

JfreeChart. http://www.jfree.org/jfreechart/

Jiménez Herrera, B; Carpio Dueñas, A. (2008) *LA CATA DE ACEITES: ACEITE DE OLIVA VIRGEN. CARACTERÍSTICAS ORGANOLÉPTICAS Y ANÁLISIS SENSORIAL. Junta de Andalucía. Instituto de Investigación y formación agraria y pesquera, Consejería de agricultura y pesca.*

Machine Learning in MATLAB. https://es.mathworks.com/help/stats/machine-learning-in-matlab.html

Majchrzak, T. Electronic noses in classification and quality control of edible oils: A review. Food Chemistry. 2018. Volume 246, 25 April, Pages 192-201

Microchip MCP23008. https://www.microchip.com/wwwproducts/en/MCP23008

Mildner-Szkudlarz, S The potential of different techniques for volatile compounds analysis coupled with PCA for the detection of the adulteration of olive oil with hazelnut oil. Food Chemistry Volume 110, Issue 3, 1 October 2008, Pages 751-761

Morales, M.T. *Comparative study of virgin olive oil sensory defects. Food Chem. 2005; 91 (2): 293-301.*

Moreno, I. (Julio 2009). *La Nariz Electrónica: Estado del Arte*

National Instruments *https://www.ni.com/data-acquisition/esa/*

NetBeans*. https://netbeans.org/*

NetBeans Developing Applications with NetBeans IDE. *https://docs.oracle.com/cd/E50453_01/doc.80/e50452/toc.htm*

Oracle Corporation. *www.java.com*

Raspberrypi Foundation. *www.raspberrypi.org*

Sensirion. The sensor company. *https://cdn-shop.adafruit.com/product-files/2857/Sensirion_Humidity_SHT3x_Datasheet_digital-767294.pdf*

Texas Instrument. ADS1115 *http://www.ti.com/lit/ds/symlink/ads1115.pdf*

The Pi4J Project*. http://pi4j.com/*

Tutorial Hostinger. *https://www.hostinger.es/tutoriales/que-es-ssh*

7. ANEXO 1: MANUAL DE INSTALACIÓN

Esta es una guía de instalación de Raspberry Pi con control remoto y el enlace con NetBeans para programar en Java con soporte Raspberry.

Instalación Raspberry Pi

1. Partimos del punto en el que tenemos una Raspberry Pi 3 B y una tarjeta SD, como mínimo debe ser de 16 GB y clase 10. La clase no es tan esencial, solo influirá en la velocidad de grabado del sistema operativo.

2. Debemos grabar el sistema operativo Raspbian, basado en Linux, en la tarjeta SD. Para ellos utilizaremos el adaptador de la tarjeta para PC.

Nos vamos a la página de Raspberry Pi. En el apartado ‘download’ descargamos el sistema Raspbian que se ajuste a nuestras necesidades. Esta descarga nos dará una imagen .iso que será la que grabemos en la tarjeta SD. Si se está utilizando una pantalla auxiliar en los pines GPIO de la RPi, podemos descargar el sistema operativo con los drivers necesarios desde el fabricante de la pantalla en concreto.

Para grabar la imagen del SO, el programa de software libre y más utilizado es el WinDisk 32. En primer lugar formatea la tarjeta SD aunque esté recién comprada. Abrimos el WinDisk 32 (Figura 7.1) y nos aseguramos que el disco seleccionado para grabar sea el correcto. Luego buscamos en los archivos

donde hayamos guardado el sistema operativo Raspbian y lo cargamos. Cuando la escritura esté lista pasamos al siguiente paso.

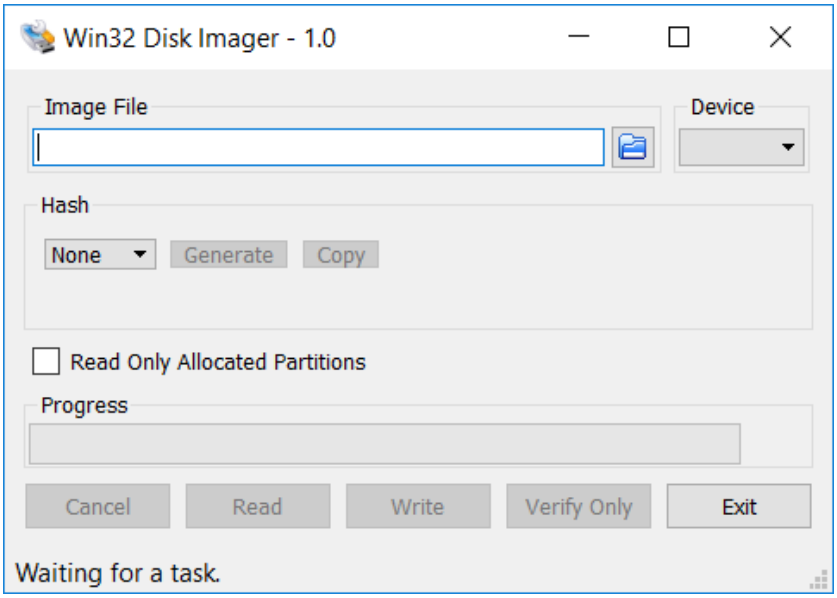


Figura 7.1 WinDisk32

3. Introducimos la microSD en la ranura habilitada en la RPi3 (Ver Figura 7.2). La RPi3 tiene 4 puertos USB u puerto HDMI. Vamos a necesitarlos para hacer la primera configuración, ya que para acceder al control remoto necesitamos primero dar otros pasos.

Conectamos la Rpi3 a la red con un transformador normal, cargador de móvil o USB. La RPi tiene una entrada microUSB. Conectamos un ratón USB y un teclado USB a los puertos de la RPi. Cuando se arranque el sistema operativo, como en un PC normal, lo más conveniente es primero configurar el teclado y el idioma de la RPi. ¿Por qué? Porque para escribir cualquier comando no podremos utilizar nuestro teclado USB. Para configurar el teclado entramos en Menu/Preferencias/Configuración de RPi/Localización/Configurar Teclado.

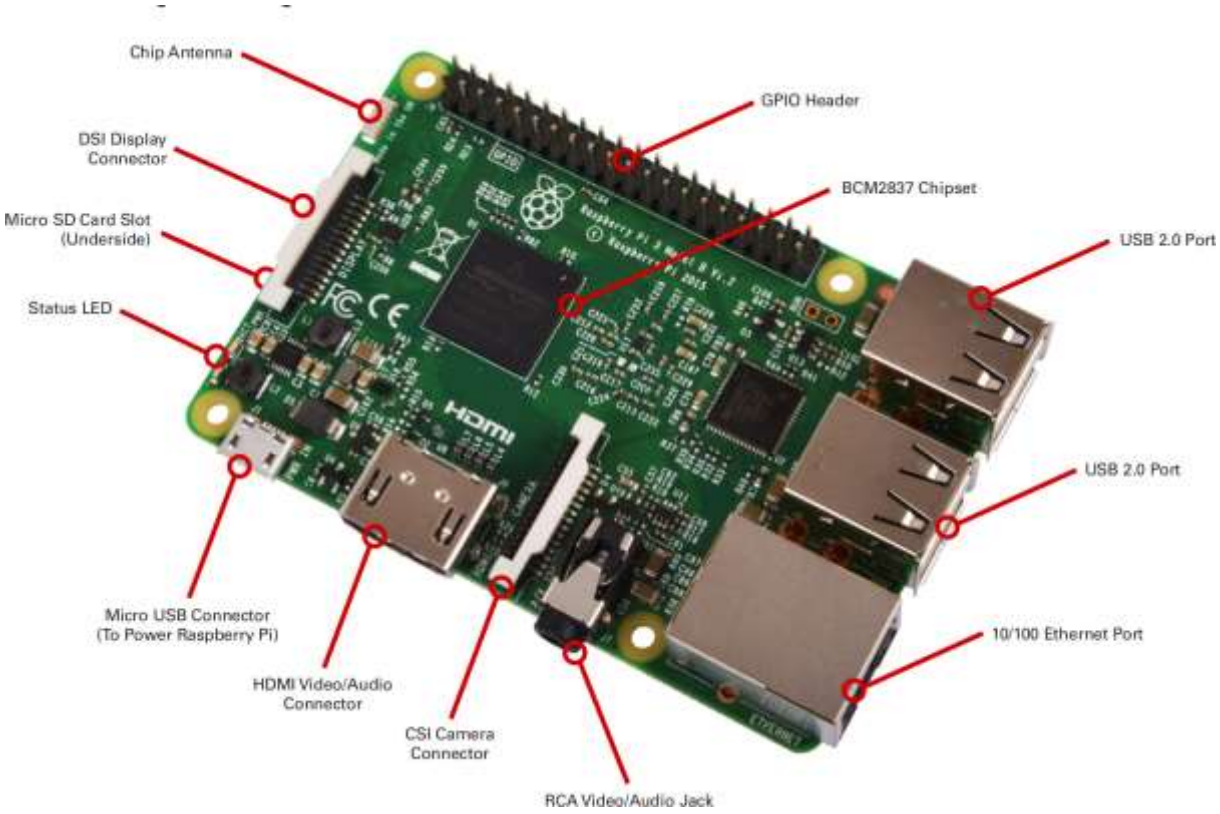


Figura 7.2 Periféricos Raspberry Pi

Una vez llegados aquí podemos elegir el país y la variante. Lo aconsejable es utilizar el Generic 105K. El idioma se puede configurar del mismo modo en Configuración Local el idioma y el país, español y España. El conjunto de caracteres aconsejable es UTF-8.

4. A partir de este punto podemos trabajar con más facilidad con la Rpi. El siguiente paso puede ser hacer configuración general de la Rpi.

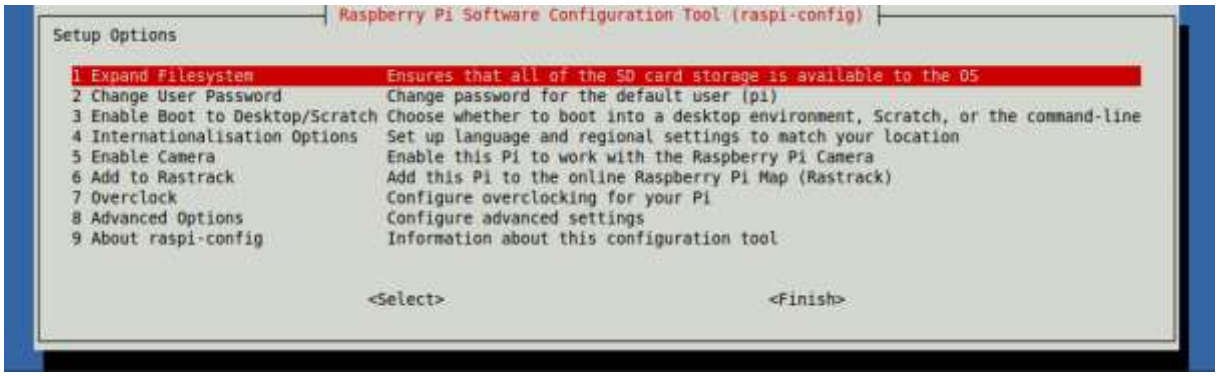


Figura 7.3 Configuración general Raspberry

5. La primera opción en la configuración general es ‘Expand Filesystem’. Debemos expandir el archivo del sistema en primer lugar antes de seguir usando la Raspberry.

6. Antes de pasar a la conexión remota, debemos activar la habilitación de comunicación SSH en la Raspberry. Esto lo encontramos en ‘Advanced Options’ de la figura 7.3.

Instalación conexión remota por SSH

7. En primer lugar debemos descargar la plataforma PuTTY de su página oficial

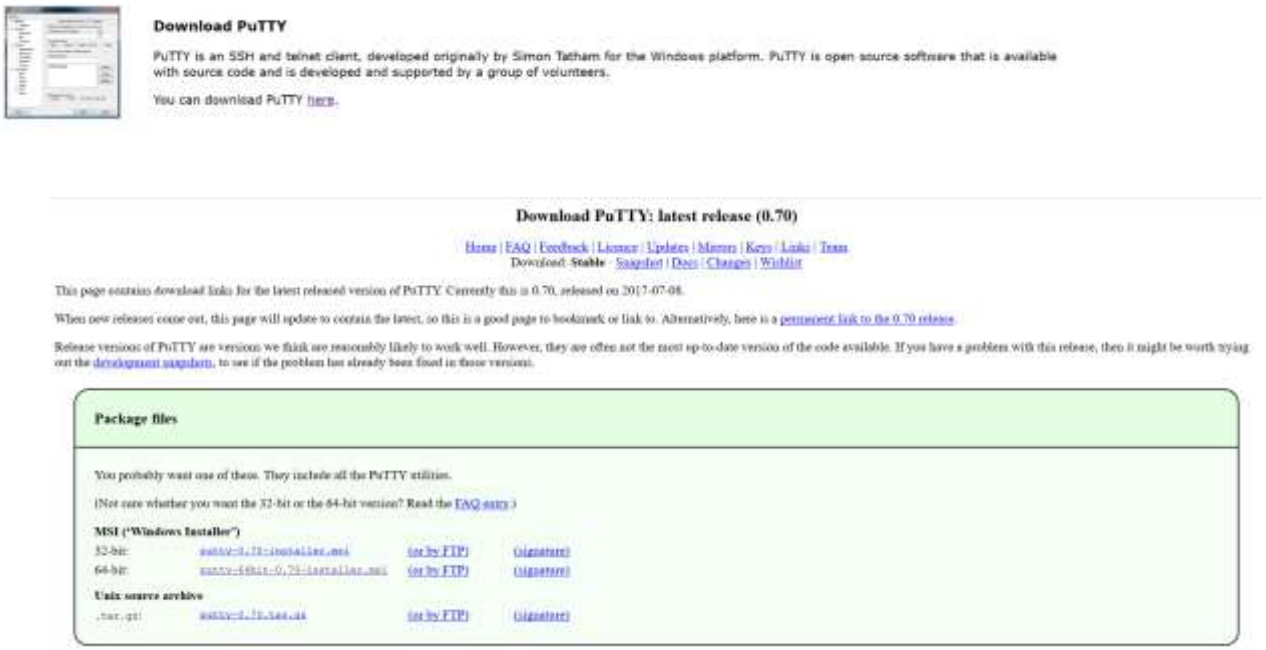


Figura 7.4 Instalación PuTTY

8. Una vez que tenemos el programa instalado en nuestro ordenador, utilizamos la dirección IP de nuestra Raspberry. Por defecto vendrá el puerto 22 configurado:

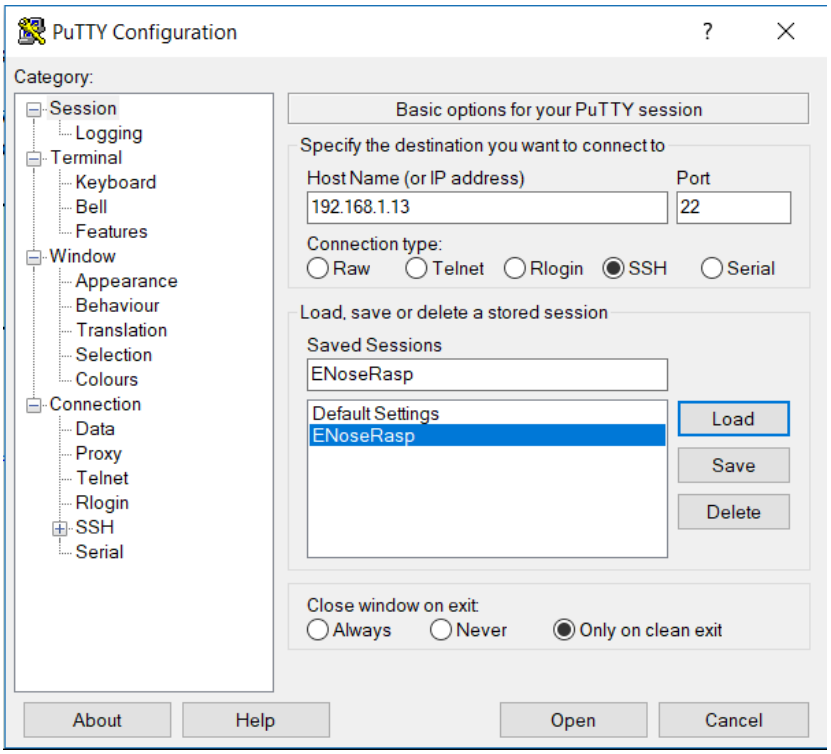


Figura 7.5 PuTTY

9. Con una conexión punto a punto entre el ordenador y la Raspberry, debemos configurar en el centro de redes y recursos compartidos, la dirección IPv4 de la conexión Ethernet. La máscara subred debe ser la misma, y la dirección IP solo debe diferir en los últimos dígitos

10. Cuando esté todo configurado, pulsamos en 'Open' y debería abrirse la siguiente pantalla. Por defecto el usuario de Raspberry es: pi. Y la contraseña es: Raspberry

Figura 7.6 Log in PuTTY 1

Figura 7.7 Log in PuTTY 2

Ya podemos controlar la Raspberry remotamente con conexión SSH, es decir, por comandos.

11. Para poder ver la pantalla de la Raspberry en nuestro ordenador y prescindir también de una pantalla supletoria, utilizamos el programa Xming. En primer lugar lo descargamos en su versión gratuita disponible para Windows. Reproducimos aquí las pantallas de instalación en las Figuras 7.8 a 7.12.

Figura 7.8 Instalación Xming 1

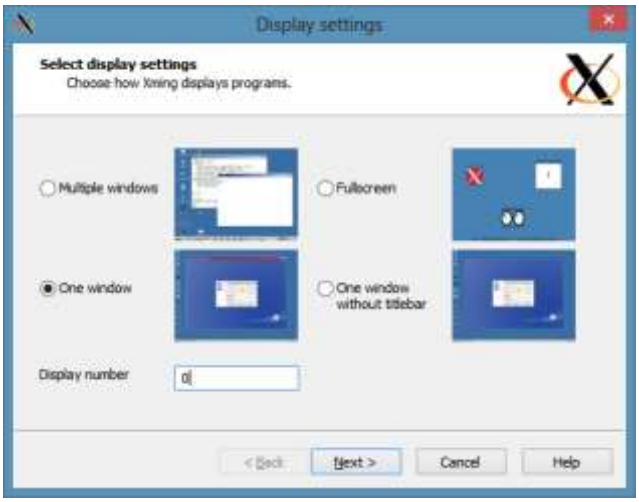


Figura 7.9 Instalación Xming 2

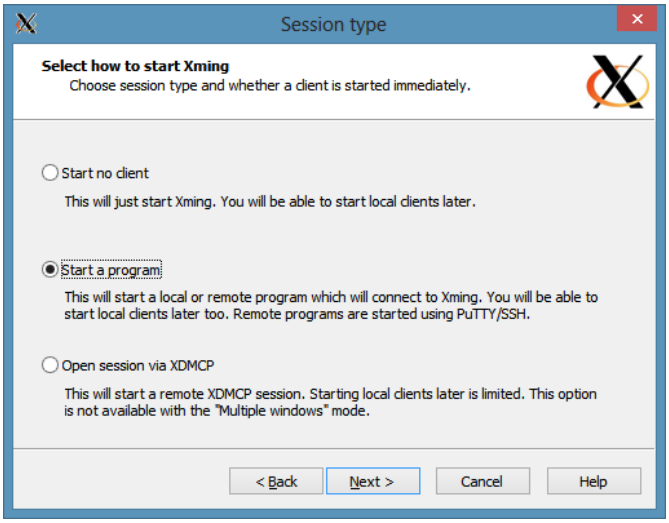


Figura 7.10 Instalación Xming 3

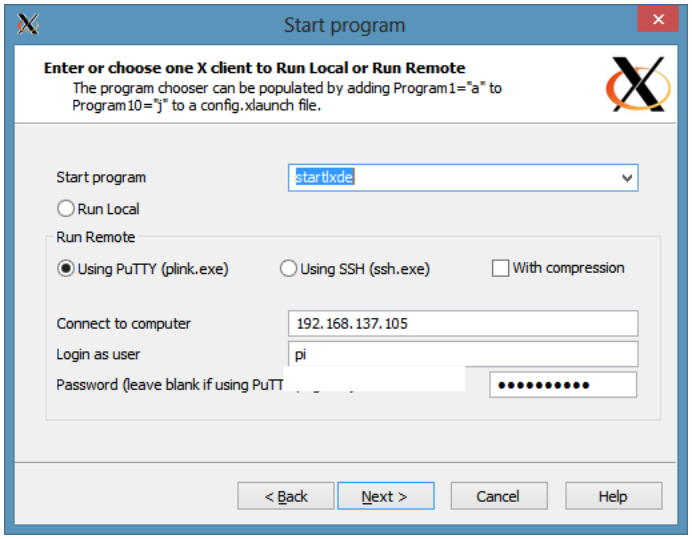


Figura 7.11 Instalación Xming 4

En el campo en blanco ‘Connect to computer’, debemos escribir la dirección IP de la Raspberry.

Una vez acabado todo este proceso, se creará un archivo .launch con el que poder ejecutar la ventana que acabamos de crear con conexión a nuestra Raspberry.

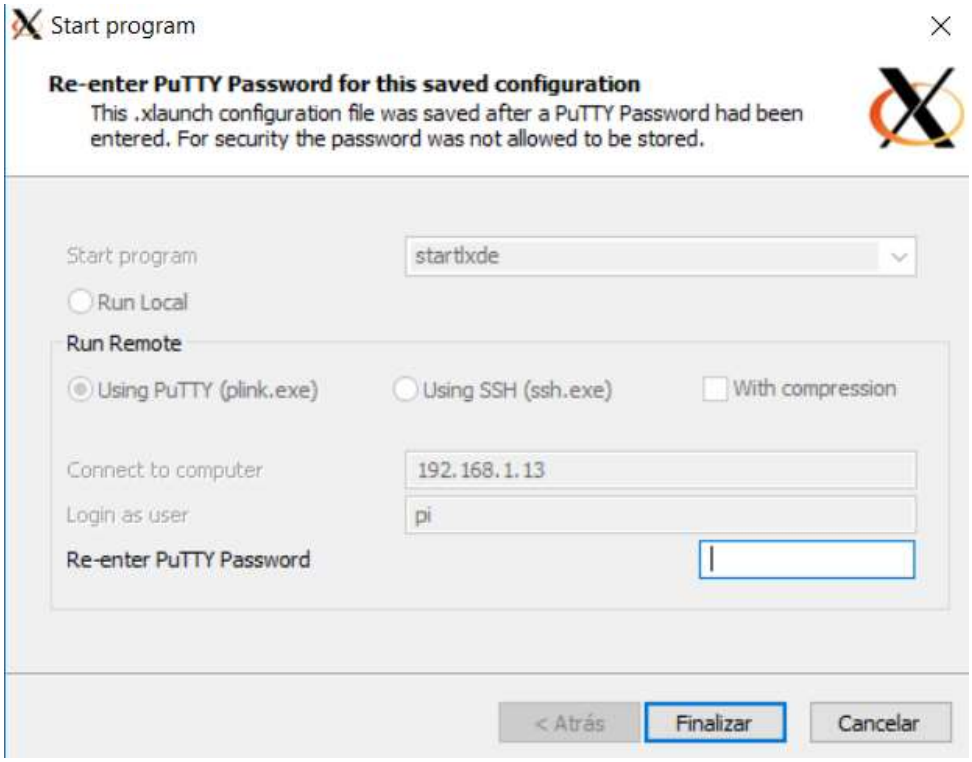


Figura 7.12 Aplicación ejecutable Xlaunch

12. Por último, siempre que queramos utilizar la Raspberry desde nuestro ordenador, debemos primero establecer la conexión con PuTTY y a continuación ejecutar la aplicación de Xming.

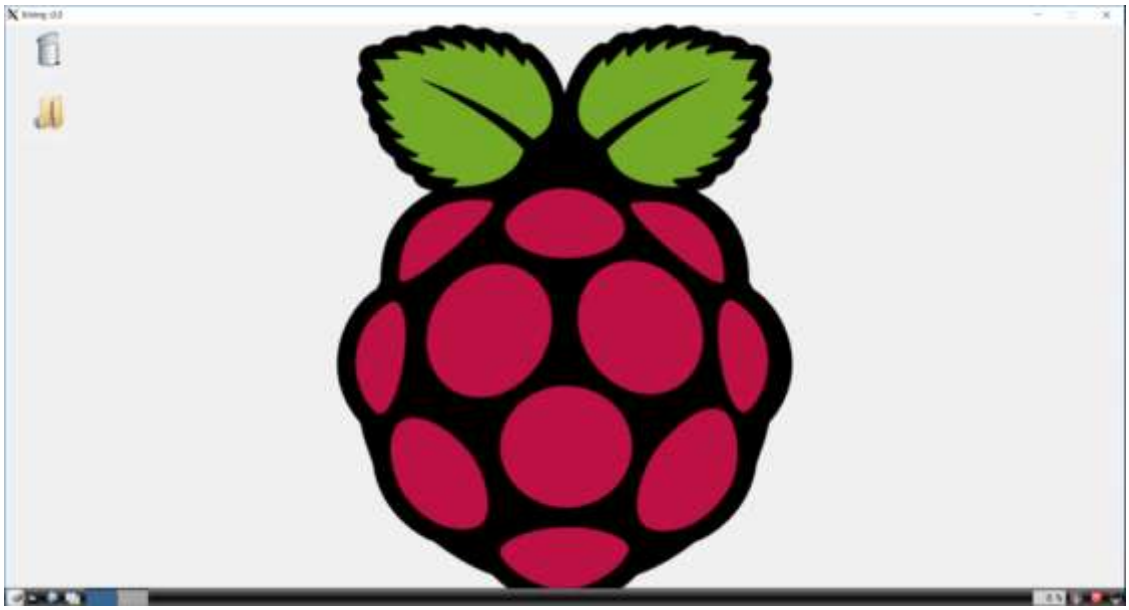


Figura 7.13 Pantalla remota Xming

Instalación Runtime Remote Platform: NetBeans y Raspberry

13. Nos vamos al entorno de programación NetBeans, en propiedades del proyecto al apartado de Build/Run.

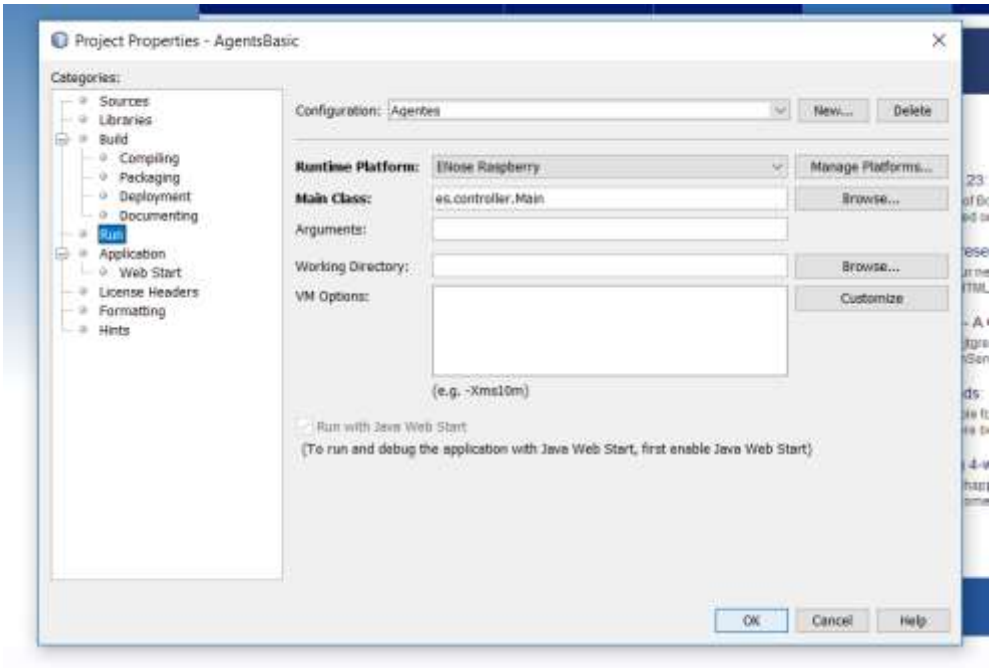


Figura 7.14 Propiedades del proyecto

Pulsamos en ‘Manage Platforms’ y nos aparecerán las plataformas de compilación que tenemos configuradas.

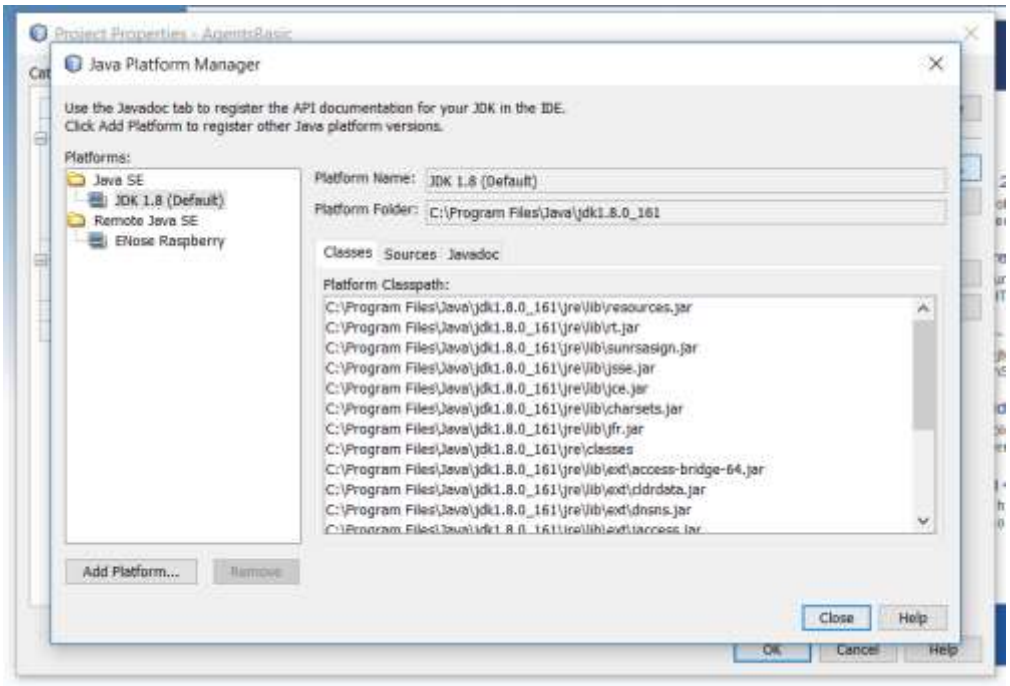


Figura 7.15 Platform Manager

En la parte inferior de la ventana pulsamos en ‘Add Platform’.

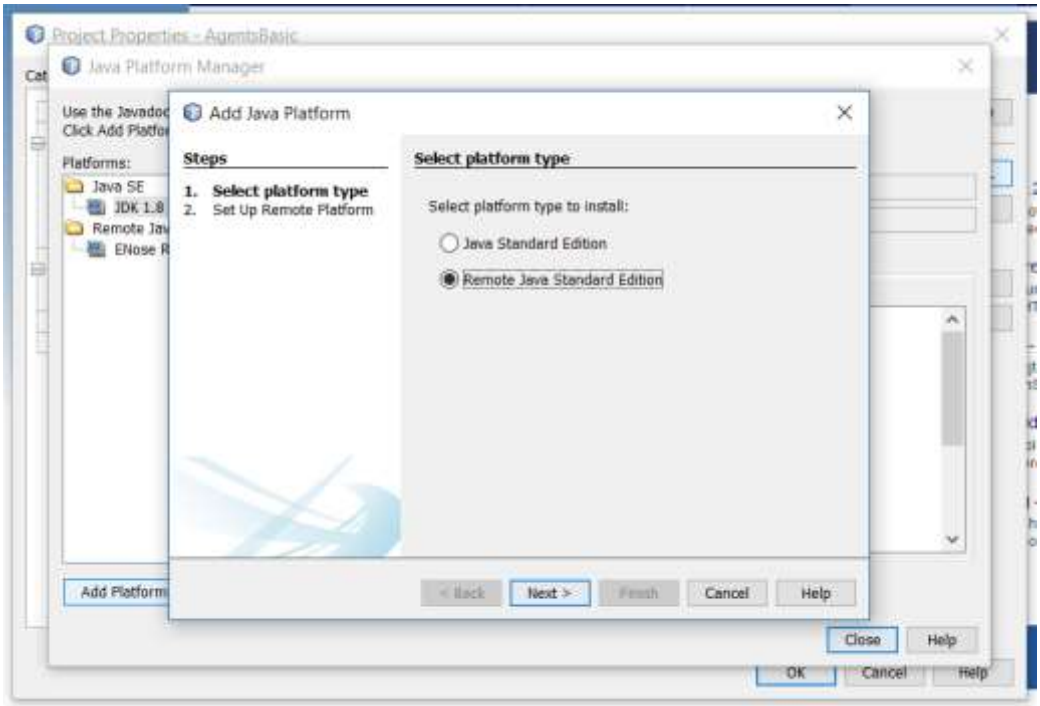


Figura 7.16 Añadir nueva plataforma remota

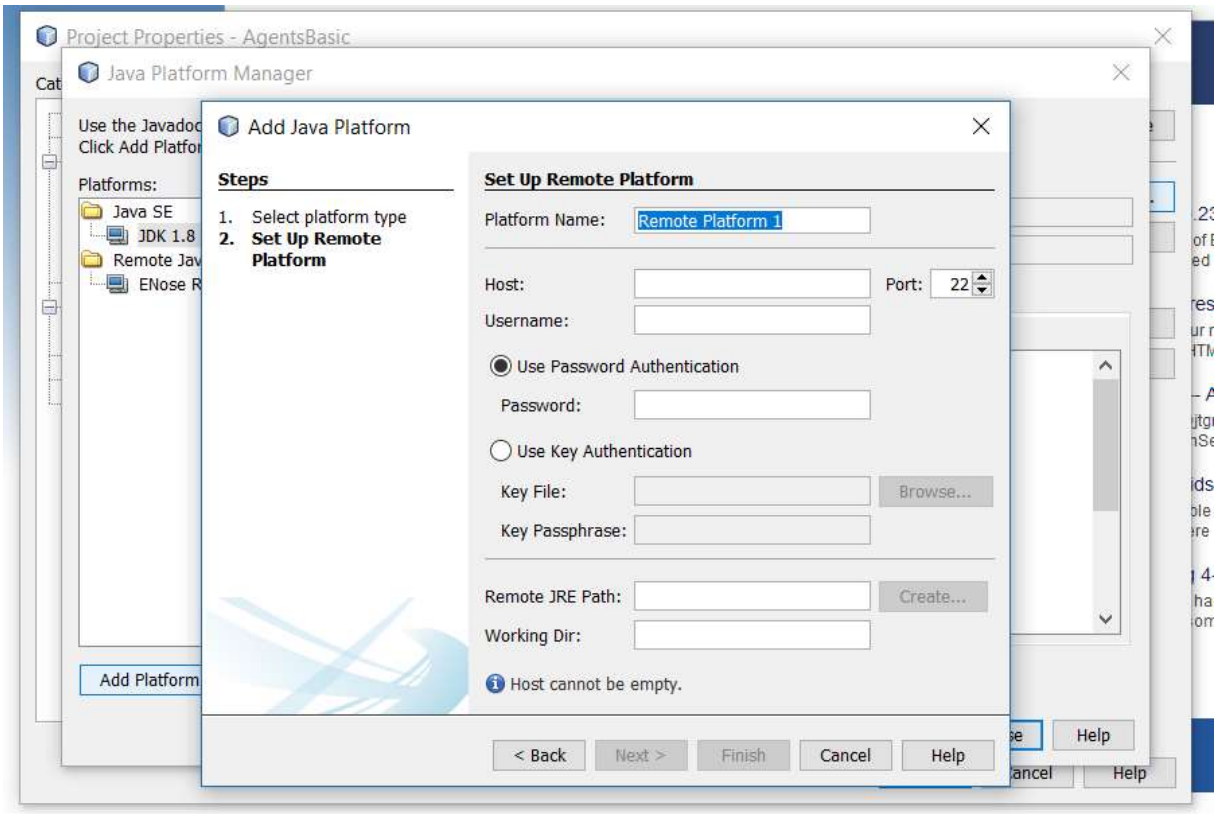


Figura 7.17 Configuración de plataforma remota 1

Introducimos los datos necesarios en la configuración. En nuestro caso eran los siguientes.

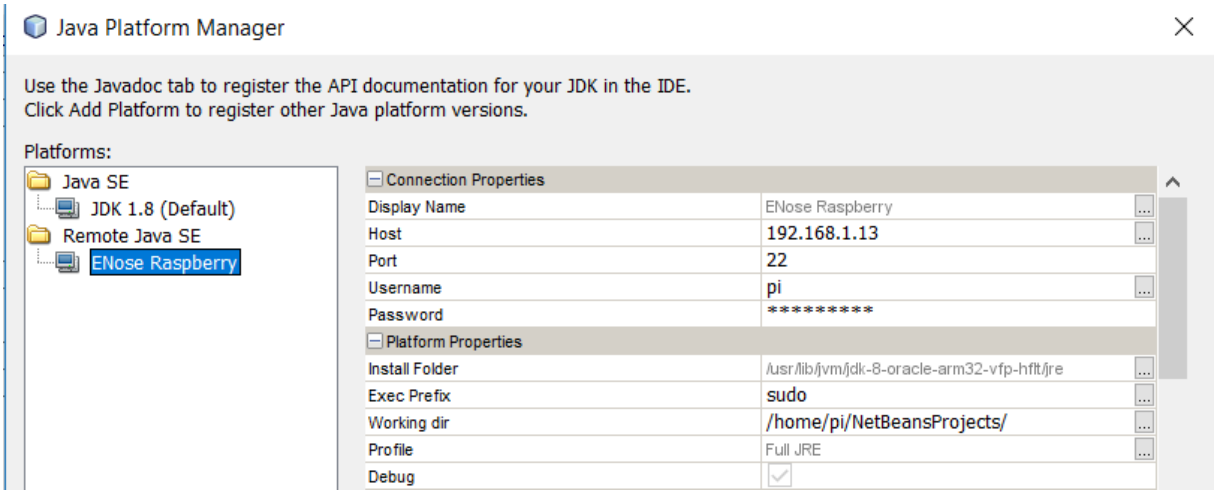


Figura 7.18 Configuración de plataforma remota 2

14. El directorio en la Raspberry Pi donde se instala el JDK 8 de Java es el ‘JRE Path’ remoto de la figura 7.14. En nuestro caso es: /usr/lib/jvm/jdk-8-oracle-arm32-vfp-hflt/jre

15. Una vez tenemos todo preparado, podemos elegir entre compilar el código Java en nuestro ordenador o en la Raspberry Pi, usando también la conexión remota SSH

8. ANEXO 2: MANUAL DE USUARIO

Ésta es una guía de usuario para el uso de la nariz electrónica.

Al encender la nariz electrónica veremos la pantalla de inicio, mostrada en la Figura 8.1.

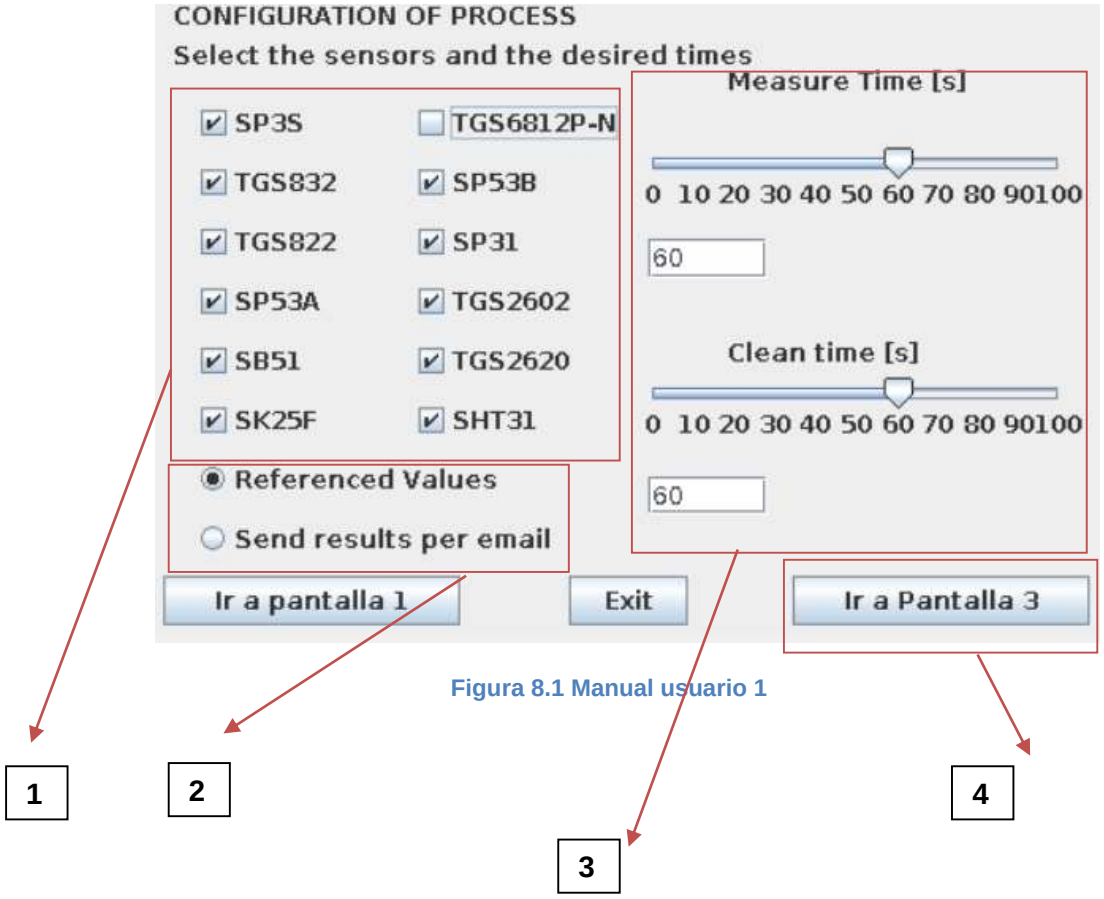


Figura 8.1 Manual usuario 1

En esta pantalla configuramos los parámetros de la medida.

1: Selección de sensores

Podemos seleccionar qué sensores queremos medir seleccionando cada uno de ellos o deseleccionándolos. Por defecto aparecerán todos seleccionados.

2: Opciones sobre el tipo de dato a representar y enviar datos por email

Si elegimos la opción ‘Referenced Values’, los datos de los sensores que se representen serán referenciados. Por defecto será esta opción la seleccionada. En caso contrario serán absolutos.

Si elegimos la opción ‘Send results per email’, aparecerá otra pantalla en la que podemos introducir un correo electrónico con el teclado táctil.

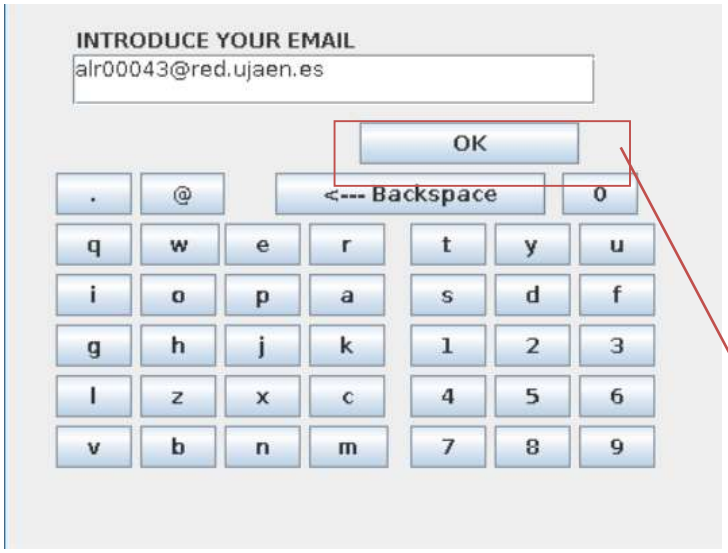


Figura 8.2 Manual de usuario 2

2.1

2.1: Al pulsar en ‘OK’ volverá automáticamente a la pantalla inicial para seguir configurando la medida.

3: Establecer tiempos de medida y limpieza deslizando la barra.

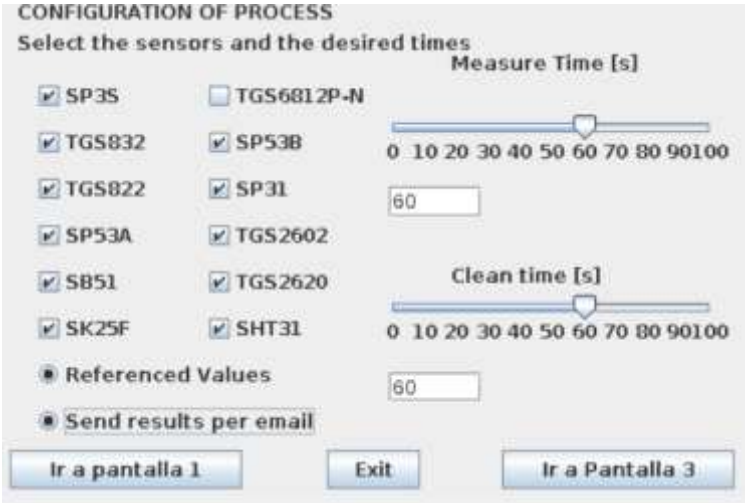


Figura 8.3 Manual de usuario 3

4: Cambiar de pantalla, tras haber configurado la medida.

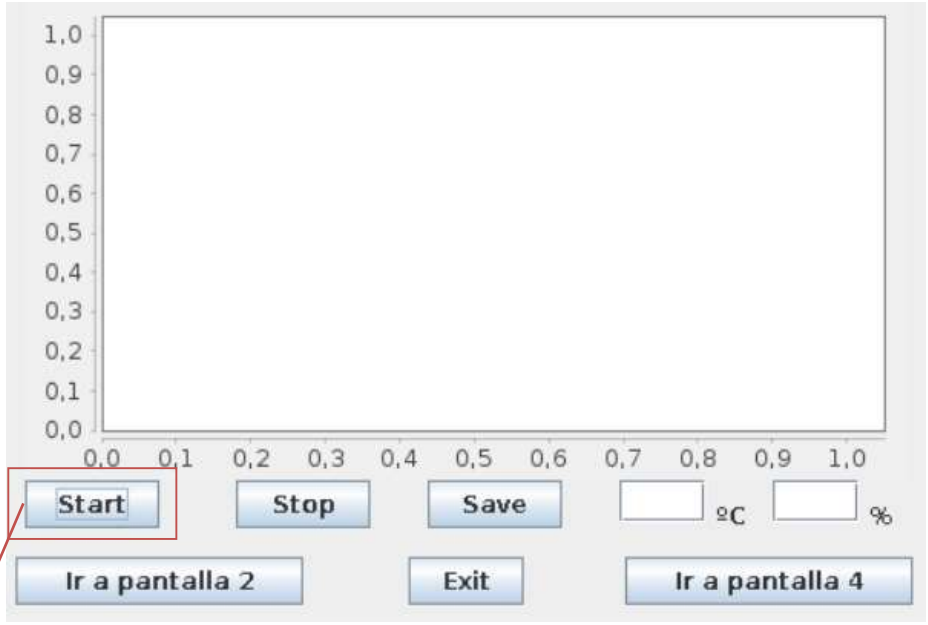


Figura 8.4 Manual usuario 4

5: Una vez que hemos cambiado de pantalla y estamos en la pantalla de visualización de la medida, hay que pulsar el botón 'Start' para que se inicie el proceso. Primero tendremos un tiempo de limpieza con el tiempo establecido. A continuación comenzará el tiempo de medida y podremos ver cómo empiezan las curvas de los sensores a graficar.

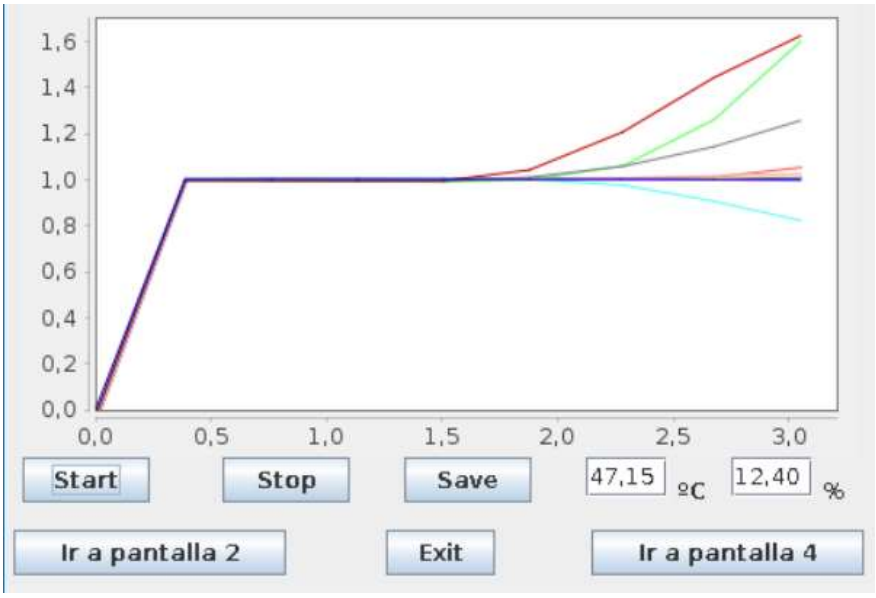


Figura 8.5 Manual usuario 5

Figura 8.6 Manual usuario 6

6: Con el botón 'Stop' se puede detener la medida en cualquier momento durante el transcurso de la representación de curvas.

7: Los valores de temperatura y humedad de la cámara de sensores los podemos ver en la parte inferior de la gráfica.

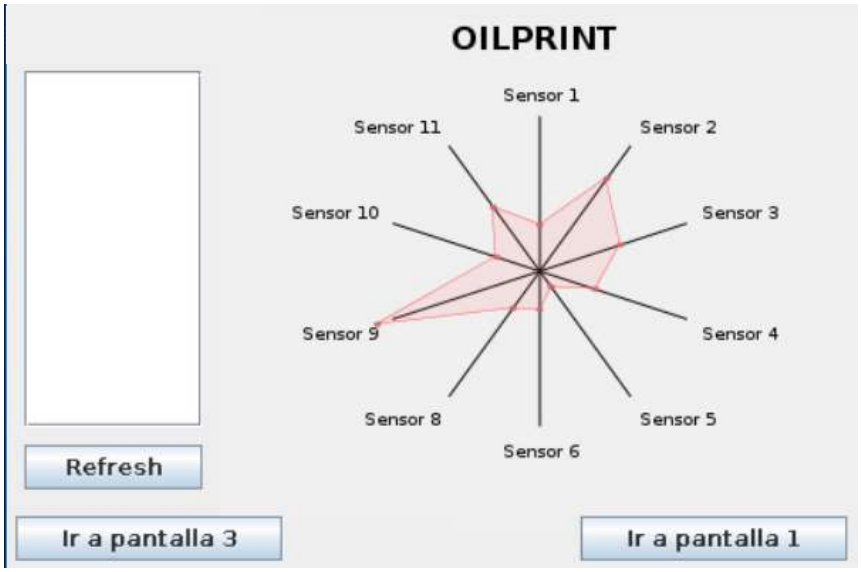


Figura 8.7 Manual usuario 7

Si cambiamos de pantalla de nuevo, podremos ver en tiempo real cómo cambia la huella aromática del aceite de la muestra.

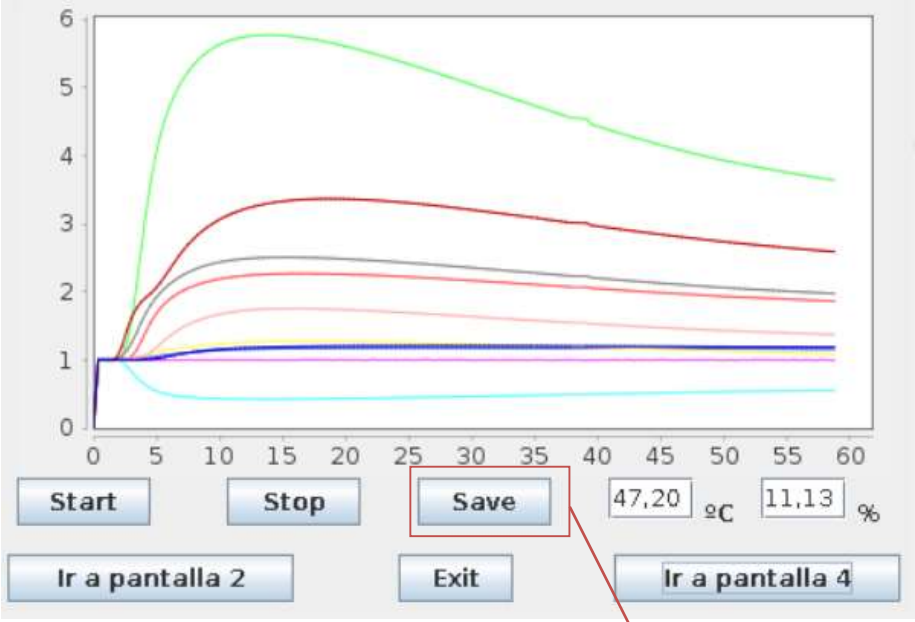


Figura 8.8 Manual usuario 8

8: Al pulsar sobre el botón de ‘Save’, los datos de la medida se guardarán en la nariz electrónica y además se enviará el correo electrónico en caso de que hayamos elegido esa opción.

Transcurrido el tiempo de medida, las curvas de los sensores habrán terminado de graficar y probablemente se hayan establecido en un régimen continuo.

La siguiente pantalla mostrará la huella característica en su valor final, del aceite que hayamos medido.

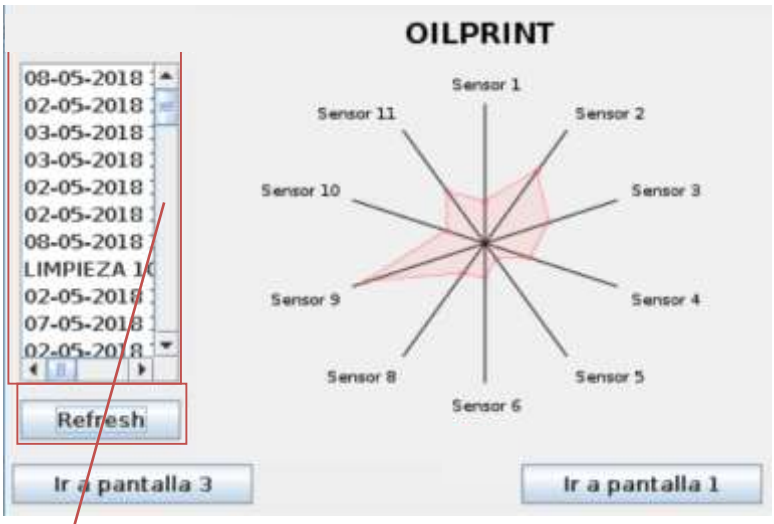


Figura 8.9 Manual usuario 9

9

10

9: El botón de 'Refresh' nos permite volver a consultar la huella característica de cualquier muestra de aceite que anteriormente se haya guardado en el histórico de medidas. Están guardadas con fecha y hora del momento en el que se tomaron. Solo con pulsar sobre ellas se podrá ver de nuevo la huella en la gráfica de la derecha.

10: La lista deslizable corresponde a todas las medidas realizadas anteriormente y se puede seleccionar cualquiera de ellas con solo pulsar sobre ellas una vez.

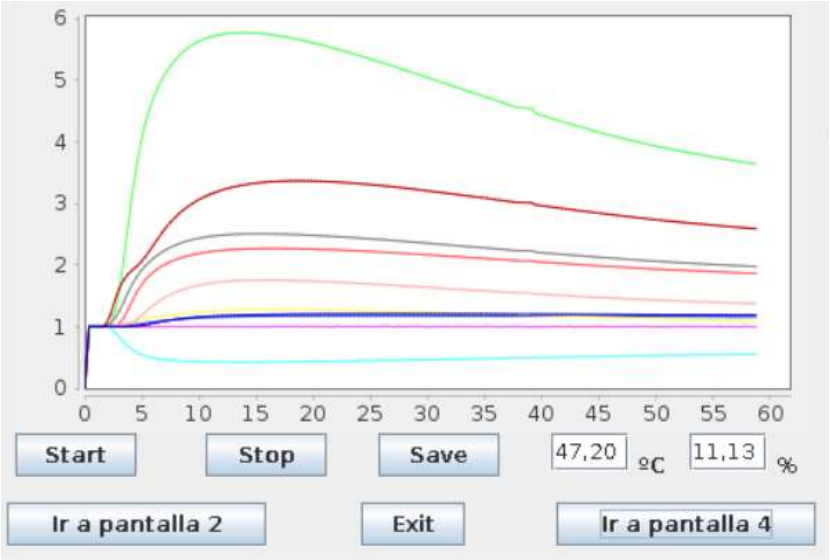


Figura 8.10 Manual de usuario 10

Para volver a la pantalla de la figura inferior, debemos pulsar sobre el botón 'Start' de nuevo para que se restablezcan las gráficas y poder iniciar un proceso de configuración de la medida otra vez.

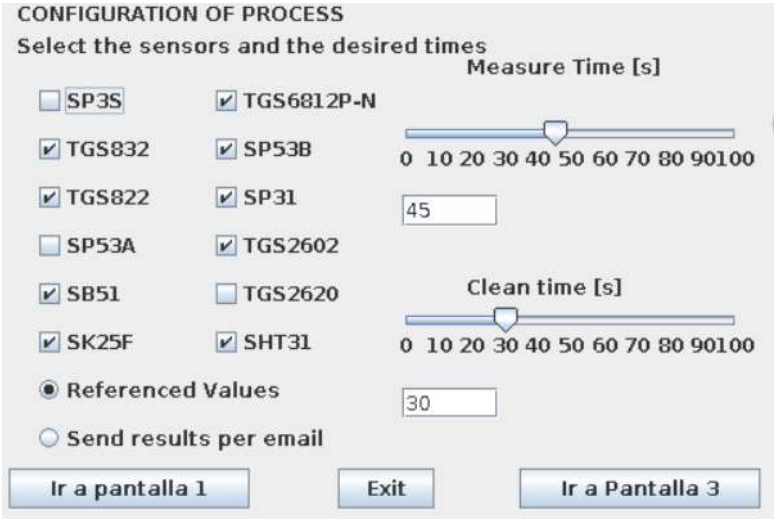


Figura 8.11 Manual usuario 11

Como se observa en la figura superior, la medida la configuramos de nuevo y volvemos a empezar el proceso que hemos seguido hasta este punto tantas veces como queramos.

Figura 8.12 Manual usuario 12

11

12

Desde la pantalla de configuración de la medida podemos cambiar a la pantalla de superusario, pulsando en ‘Ir a pantalla 1’.

11: El estado de los sensores se puede comprobar en esta parte de la pantalla.

12: El botón ‘Request’ recarga todos los datos de cada sensor cada vez que se pulsa.