eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
ZIENTZIA ETA TEKNOLOGIA FAKULTATEA
FACULTAD DE CIENCIA Y TECNOLOGÍA

Trabajo Fin de Grado
Grado en Ingeniería Electrónica

Estandarización de Interfaz para Test Electrónico

Diseño, programación e implementación del sistema

Autor:
Jon Martínez Sánchez
Director:
Iker Caballero Ortiz de Zárate

Leioa, 19 de Junio de 2019

Índice

1-Introducción y objetivos del trabajo

En la industria de la electrónica, existe, como probablemente en todas las industrias que se dediquen a la fabricación de un producto, una etapa de control de calidad. Lo que se hace en ésta etapa es, básicamente, probar que los productos en la cadena de producción se fabrican sin errores. En la gran mayoría de los casos, los fallos provocados en el producto en cuestión no permiten su distribución comercial, y si no se mantiene una atención constante en el dicho control de calidad, se podrían perder importantes sumas de dinero.

En el contexto del presente trabajo, el objetivo del mismo es realizar un estándar de un sistema para test electrónico ICT, que sea fácilmente programable por el operario que se haga cargo del mismo. Otro objetivo que también persigue este trabajo es, como su nombre indica, realizar un sistema funcionalmente flexible, es decir, que se adapte a la mayor cantidad de sistemas a testear posible.

1.1-Principales tipos de test

En el caso de la industria electrónica, se distinguen dos tipos de test: el test funcional (FCT[1]), y el test electrónico (ICT[2]). Estos tipos de test también se llevan a cabo frecuentemente cuando se está desarrollando un software, y se quiere probar tanto su funcionalidad como la eficiencia del código en sí.

1.1.1-Test funcional

Como su nombre puede dar a entender, este tipo de pruebas normalmente no tiene en cuenta la electrónica, ni en general, los principios físicos que rigen el funcionamiento del aparato cuya funcionalidad se quiere probar. Así que, en definitiva, se trata de un test en el que se comprueba que el producto está preparado para su uso mediante un caso de uso real en el que se testean todas sus funcionalidades.

1.1.2-Test electrónico

Es un test en el que se prueba que la electrónica no falla. A menudo, se miden todos los componentes del circuito uno por uno, pudiendo así detectar fallos imperceptibles mediante un test funcional. De esta forma, podemos detectar errores que pudieran dar lugar a una correcta funcionalidad, pero grandes limitaciones en la durabilidad del sistema u otros errores que, de una forma de otra, no permiten un correcto funcionamiento del dispositivo electrónico que se testea. Cabe destacar que el test electrónico suele realizarse previamente al test funcional, para evitar daños mayores (tales como los causados debido a una incorrecta orientación de condensadores electrolíticos o diodos zener). [1]

Algunos otros errores que pueden surgir en los circuitos electrónicos, pero que forman parte de comprobaciones previas, son los referidos al montaje físico, es decir, errores de fabricación, como puede ser la inexistencia de algún componente en una posición determinada, cortocircuitos, …

[1] FCT: *Functional Test*.
[2] ICT: *In Circuit Test.*

Con respecto a los tipos de señales que se miden en un test electrónico, fundamentalmente son las mismas que en un multímetro: voltaje, intensidad y resistencia, tanto si las señales que se van a medir son analógicas como si son digitales. [2]

Aun así, existen algunas mediciones adicionales que se pueden llevar a cabo para completar la prueba, como pueden ser medidores de luminosidad y color de LEDs, medidores de frecuencia y de temperatura, pudiendo así evaluar más parámetros de los sistemas bajo test.

1.2-Tipos de interfaz usadas en ICT

Con respecto a la arquitectura de los dispositivos de test electrónico, existen varias interfaces. Una de ellas sería el test manual, en la que el contacto con los pines se realiza, como su propio nombre indica, manualmente a través de sondas. Entre las interfaces automatizadas, destacan dos tipos: *Bed of Nails* y *Flying Probe Test*. En cualquiera de los casos, se utiliza una gran variedad de puntas de test, y que normalmente se clasifican según su geometría o el uso que se las dé. El fabricante **Feinmetall** incluye la siguiente ilustración en su catálogo, clasificándolas según el uso:

Ilustración 1: algunas de las puntas para test electrónico ofrecidas por el fabricante Feinmetall. (catálogo de puntas de test de 2017). [3]

1.2.1-Bed of Nails

Esta interfaz recibe su nombre por su estructura basada en una matriz de puntas de test, que hacen contacto con los nodos del circuito cuyas características queremos medir. Para conectar el circuito y las puntas de test, existen tres métodos generales: utilizar una bomba de vacío (aunque requiere que el PCB no tenga agujeros para ser efectivo), usar una fuente de aire comprimido, o bien un sistema mecánico. [4]

1.2.2-Flying Probe Test

Es un tipo de test más dinámico y que se ajusta a una mayor cantidad de componentes electrónicos. Las puntas se mueven mediante un sistema electromecánico, y se utilizan dos o más puntas simultáneamente, controladas vía software. Además, el sistema es menos propenso a causar desperfectos en los circuitos que se evalúan.

Por otra parte, el sistema es mucho más caro en éste caso, dada la complejidad de la maquinaria electromecánica y el sistema de control, y da lugar también a un test más lento, ya que las puntas de test hacen los contactos con los nodos y elementos del circuito de forma secuencial.

1.3-Objetivos del trabajo

El presente proyecto se basará en la implementación y programación (abarcando así, tanto software como hardware) de un sistema de test que sea lo más generalista posible, y que al mismo tiempo tenga capacidad de realizar una amplia serie de medidas. Para su programación, procesado, y exposición de los resultados, se utilizará una tarjeta de desarrollo basada en un microcontrolador (en este caso, la tarjeta **Arduino MEGA 2560 R3**). Se presenta a continuación, una serie de objetivos a cumplir:

·Medida de varias magnitudes físicas: para caracterizar el sistema bajo test, se realizarán medidas de tensión tanto analógicas como digitales, temperatura, señales acústicas, medidas de frecuencia, luminosidad y color. También será necesario generar señales de tensión analógica y digital.

·Fiabilidad en las medidas: es decir, un aparato que consiga realizar medidas con un margen de error relativamente pequeño.

·Sencillez en su programación: es importante hacer que el sistema de medida sea fácilmente programable, y que no requiera de unos conocimientos avanzados en informática y electrónica. Las secuencias de test que el sistema lleve a cabo han de ser fácilmente programables, a través de un código de instrucciones genérico con una estructura estándar, y si es posible, de longitud fija.

·Sistema flexible y adaptativo: el objetivo principal del proyecto es crear un método de test que pueda poner a prueba la mayor cantidad de dispositivos posible, sin tener que realizar grandes cambios en el mismo.

·Interfaz de usuario intuitiva: además de una programación sencilla, también constará de una interfaz de usuario clara y concisa, en la que los resultados de los test que se realicen sean fácilmente interpretables.

2-Diseño a alto nivel

El sistema a implementar, ha de tener medios suficientes para medir los tipos de señales mencionados en la introducción del presente proyecto, así como de gestionar sus datos y mostrarlos por pantalla. Se expondrán a continuación, desde una perspectiva teórica, los bloques componentes necesarios para implementar el sistema de test, para que responda al siguiente diagrama de bloques:

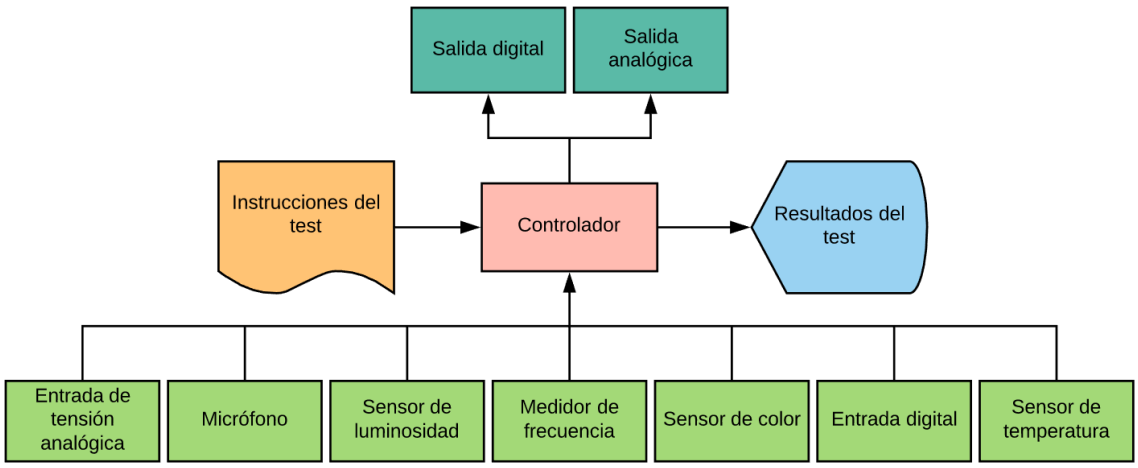

Ilustración 2: diagrama de bloques funcional del sistema de test.

Cabe aclarar que el diagrama mostrado es conceptual, y no siempre será necesario un instrumento adicional al controlador para cumplir alguna función determinada. Concretamente, en el sistema que se quiere implementar, no es necesario instalar medidores de frecuencia, ni dispositivos que realicen lecturas e interpretaciones de señales de tensión continua, ni de señales digitales. Tampoco hará falta un dispositivo que emita señales digitales; dichas tareas pueden ser realizadas directamente por el controlador, que en este caso se refiere a la tarjeta de desarrollo.

2.1-DAC R-2R

La modulación por ancho de pulso (PWM[3]) es una estrategia que trata de generar señales cuyo valor medio en un periodo. Esta técnica es comúnmente utilizada en servomotores y conversores analógico-digital, entre otros, y es de hecho, la que utilizan las tarjetas de desarrollo que forman parte del proyecto **Arduino** en algunos de sus puertos. [5]

La técnica PWM destaca por su simplicidad, ya que, para variar el valor medio, basta con cambiar el tiempo que la señal digital está en nivel alto. Sin embargo, puede resultar perjudicial para los

[3] PWM: *Pulse Width Modulation*.

sistemas bajo test, ya que la amplitud de dicha señal PWM puede ser superior a la admitida por el dispositivo bajo pruebas. Existen entonces, dos alternativas sencillas:

·Filtro paso bajo: consiste en filtrar la señal PWM para obtener una señal de tensión continua. Requiere un diseño cuidadoso, ya que pueden filtrarse armónicos superiores, y además su respuesta es lenta.

·DAC R-2R: es probablemente el conversor analógico-digital más simple. La arquitectura genérica de este DAC[4] se basa en una red de resistencias como la mostrada a continuación.

Frecuentemente, estas redes de resistencias se conectan a circuitos amplificadores no inversores, o como es el caso, a circuitos seguidores de tensión tal y como se muestra en la ilustración 3. Nótese que se han sustituido los interruptores por fuentes de tensión generadoras de tensión rectangular, pero que tienen el mismo efecto como conmutador. La fuente $\mathbf{V_7}$ es la de menor frecuencia, y cada una de las siguientes hacia su izquierda es de frecuencia doble a la anterior, siendo $\mathbf{V_0}$ la de mayor frecuencia.

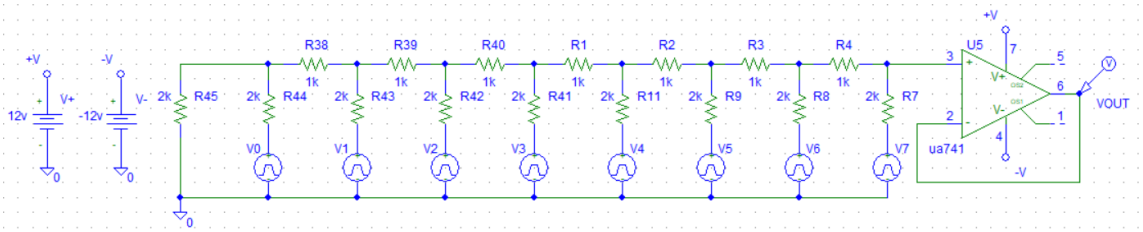

Ilustración 3: esquema eléctrico de un DAC R-2R de 8 bits. [6]

Para entender el funcionamiento del circuito mostrado en la figura anterior, resulta sencillo hacer uso explícito de los teoremas de superposición y de **Thevenin**. [7]

Puede demostrarse, que para un DAC R-2R de **n** bits, el voltaje a su salida puede generalizarse por la siguiente expresión:

$$V_O = V \cdot \sum_{i=0}^{n-1} \frac{b_i}{2^{n-i}} \qquad \textit{(Ec. 1)}$$

Donde $\mathbf{b_i}$ es una variable discreta que solo puede tomar los valores 0 o 1, representando así que el *switch* con el que comparte su subíndice está conectada ($\mathbf{b_i}$=**1**) o no ($\mathbf{b_i}$=**0**). El número de bits se representa por **n**, y **V** es la tensión de alimentación. Así, b_0 se asociaría con el interruptor relacionado que se abre o se cierra con el bit de menor peso, y al revés con el *switch* situado más a la derecha del todo (b_{n-1}).

Obtenemos, utilizando la fórmula para el voltaje saliente del conversor, que su valor ha de estar entre 0 voltios y 4.9805 voltios, aproximadamente, si el voltaje que se conecta a los interruptores es de 5 V. El voltaje máximo puede obtenerse de forma general mediante la siguiente expresión:

[4] DAC: *Digital to Analog Converter*.

$$V_{max} = V \cdot \frac{2^{n-1}}{2^n} \qquad \textit{(Ec. 2)}$$

Donde, para este caso particular, **n**=8, **V**=5. Si se realiza la simulación en **PSPICE**, el voltaje a la salida del conversor es el siguiente:

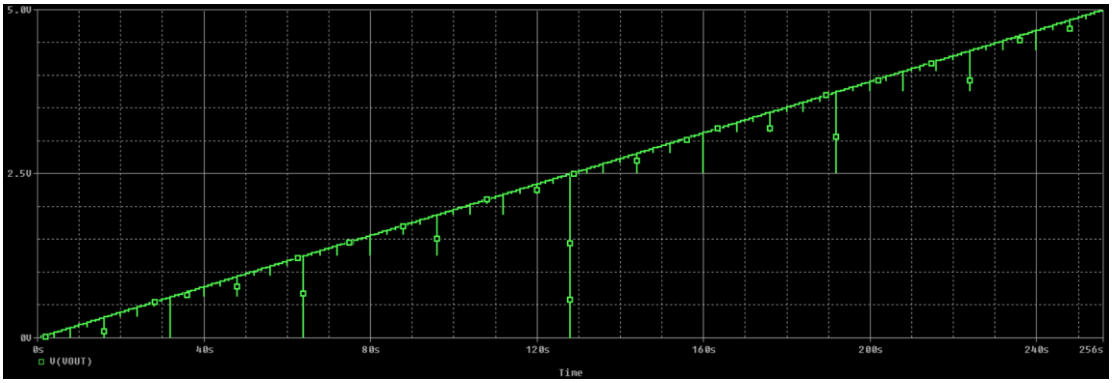

Ilustración 4: resultado de la simulación para el conversor de 8 bits.

2.2-Sensores utilizados para la medida de otras magnitudes físicas

Se explicará, a continuación, el fundamento de los sensores que se van a utilizar a la hora de implementar el hardware del proyecto, así como su finalidad en el mismo.

2.2.1-Fotorresistencia

Las fotorresistencias (también conocidas popularmente como LDR[5]), son elementos electrónicos pasivos que se caracterizan por aportar una resistencia variable con la luz que incide sobre las mismas. El efecto que da lugar a dichos cambios en la conductividad del material es el conocido efecto fotoeléctrico. Esta magnitud es variable, por tanto, mediante radiación óptica. La energía asociada a este último factor puede describirse mediante la siguiente ecuación: [8]

$$E = hf \qquad \textit{(Ec. 3)}$$

Donde **E** es la energía, **h**[6] es la **Constante de Planck** y **f** es la frecuencia. Este efecto afecta a la fotorresistencia en el momento en el que la radiación incidente porte suficiente energía como para permitir el salto de los electrones desde la banda de valencia hasta la banda de conducción. Para el caso que nos ocupa, puede ser conveniente expresar la ecuación anterior de la siguiente forma: [8]

$$\lambda(\mu m) = \frac{1.24}{E(eV)} \qquad \textit{(Ec. 4)}$$

Donde se muestra una relación inversamente proporcional entre la energía aportada a los electrones en el semiconductor y la longitud de onda de la radiación que llega hasta la LDR. Estos elementos son utilizados comúnmente como sensores de luminosidad (como será el caso), aunque pueden ser también utilizados como detectores de colores, debido a la relación mostrada en la ecuación (4). [8]

[5] LDR: *Light Dependent Resistor*.

[6] **h**=6.626·10^{-34} J·s, es la Constante de Planck.

Ilustración 5: fotorresistencia del fabricante Luna Optoelectronics. [9]

2.2.2-Micrófono electret

Los micrófonos de condensador constan de cinco piezas fundamentales: una rejilla o malla, un diafragma, una cubierta, una placa metálica trasera y una capa aislante.

Este tipo de condensadores se polariza a través de una fuente externa de tensión continua o a través de una carga eléctrica directamente aplicada sobre el material aislante. Cuando las ondas de presión relacionadas con el sonido que queremos medir fluctúan, la distancia entre el diafragma y la placa metálica situada a continuación varía. Así, también cambia la capacidad del condensador formado por ambas placas y como la carga en el condensador se mantiene constante, lo que varía es la tensión de salida del micrófono. [10]

El parámetro que afecta en mayor medida a la sensibilidad es el tamaño del diafragma, y cuanto mayor es su sensibilidad, mayor afecta a esta característica del micrófono. El rango de frecuencias operativo del micrófono depende del tamaño del mismo, y a menor tamaño, mayor es el rango de frecuencias ante los que ofrece una respuesta. [10]

De cualquier manera, el uso del micrófono en este proyecto será “digital”, es decir, se comprobará si algún sonido se ha generado o no.

A menudo, algunos aparatos electrónicos notifican al usuario sobre su estado a través de señales acústicas. Por ejemplo, muchos ordenadores cuentan con un *buzzer* interno. Mediante dicho *buzzer*, haciendo uso de combinaciones de pitidos largos y cortos, informan acerca del estado de la RAM, la BIOS, controladores de teclado, y una extensa lista de parámetros de la arquitectura del computador. El micrófono instalado en el sistema estaría destinado a comprobar que dichos generadores de señales acústicas funcionan.

2.2.3-NTC

Dentro de los sensores de temperatura, existen cuatro alternativas principales:

·Los termopares, que destacan por sus resultados precisos y características temperatura-tensión altamente lineales.

·Las RTD[7], que se distinguen por un rango lineal amplio y su estabilidad.

[7] RTD: *Resistance Temperature Detector*. Se denominan así los sensores de temperatura resistivos en los que la temperatura afecta notablemente a la conductividad de un elemento conductor. Usualmente se fabrican en platino.

·Las PTC[8], que junto con las NTC, entran en el grupo de los termistores y se basan en la variación de la conductividad de un semiconductor con la temperatura.

·Las NTC, que serán las que se utilicen en este caso para una medida efectiva de la temperatura.

Las siglas **NTC** vienen de **Negative Temperature Coefficient**, y su nombre es debido a que la dependencia de la resistencia de estos elementos varía con la temperatura de forma proporcional a una exponencial negativa. Los aumentos en la temperatura del material (habitualmente, óxidos dopados) generan aumentos en el número de portadores, causando así una disminución en su resistencia (de ahí la forma de sus características temperatura-resistencia).

Ilustración 6: NTC del fabricante EPCOS, modelo B57164K0103J000. [11]

La ecuación que gobierna la resistencia de estos elementos es la siguiente (para un rango de temperaturas limitado):

$$R_T = R_0 \cdot e^{\beta \cdot \left(\frac{1}{T} - \frac{1}{T_0}\right)} \qquad \textit{(Ec. 5)}$$

Donde R_0 es la resistencia medida a una temperatura de referencia, T_0 es dicha temperatura, y β es un coeficiente que determina la variación de resistencia con la temperatura, y que suele rondar entre los 2000 y los 5000 Kº. A menudo, también se utiliza la ecuación de **Steinhart-Hart**: [12]

$$\frac{1}{T} = b_0 + b_1 \cdot ln(R_T) + b_3 \cdot \left(ln(R_T)\right)^3 \qquad \textit{(Ec. 6)}$$

Nótese que en la ecuación existe un coeficiente b_3 pero no un coeficiente b_2; esto es debido a que sus autores probaron que dicho término (existente en una ecuación anterior) no suponía una pérdida de precisión notoria. Nótese que ambas expresiones dan resultados en ºK, con lo que es necesaria una posterior conversión de unidades a ºC.

En ambos casos, los termistores que se van a utilizar estarán situados en un divisor de tensión. La NTC de la marca cuyo coeficiente β se conoce, se conectará entre el nodo de salida y tierra, con la resistencia pasiva conectada entre la alimentación y el nodo de salida. La NTC de la que se conocen los coeficientes de Steinhart-Hart irá conectada del mismo modo, pero intercambiando las posiciones de la resistencia activa y la pasiva. Así, podrá seguirse el manual

[8] PTC son siglas de *Positive Temperature Coefficient*.

adjunto en el kit en el que también se encontraba dicha NTC, que incluye además, el código necesario. [13]

Puede deducirse, que la ecuación para la temperatura, es, para la NTC con coeficiente β:

$$T(^{o}C) = \frac{1}{\frac{1}{T_0} + \frac{1}{\beta} \cdot ln\left(\frac{R}{R_0} \cdot \frac{V_o}{V_i - V_o}\right)} - 273.15 \qquad \textit{(Ec. 7)}$$

La elección de la NTC como sensor de temperatura está justificada por su bajo coste, por su sensibilidad y por su disponibilidad en el mercado. Las características de estos elementos hacen que la relación entre temperatura y resistencia sea fuertemente no-lineal, pero no es ese el mayor de los problemas cuando se dispone de un microcontrolador que puede realizar los cálculos necesarios. [14]

De cara al proyecto, la función que desempeñarán será medir temperaturas en el dispositivo bajo test. Es de gran importancia, ya que, probablemente, en todos los dispositivos electrónicos del mercado se dan unas especificaciones de temperatura que han de respetarse para su correcto funcionamiento.

2.2.4-Fototransistor

Existe una variedad de elementos de circuito no-lineales dedicada a la medida de variables ópticas, entre los cuales, destacan LDRs, fotodiodos y fototransistores. Al igual que las LDR, los fotodiodos y fototransistores están compuestos por semiconductores que relacionan la radiación electromagnética incidente sobre los mismos con una corriente generada.

Tanto en fotodiodos como en fototransistores, aparece una magnitud llamada *corriente de oscuridad*. Este parámetro es, como su nombre indica, una corriente generada por fluctuaciones aleatorias de portadores y huecos cuando no está expuesto a ninguna fuente de luz, y que, por lo general, es dependiente de la temperatura a la que se encuentra. Interesa por lo general que el valor de esta corriente sea pequeño. [15]

El funcionamiento del fototransistor es similar al del fotodiodo, pero el transistor (por el simple hecho de ser un transistor) tiene la capacidad de amplificar la corriente de base, que en este caso es generada por el efecto fotoeléctrico. Algunos modelos cuentan con tres terminales, de forma que cubriendo su encapsulado con un material opaco, pasa a funcionar como un transistor BJT NPN común.

Ilustración 7: fototransistor SFH 310-2/3 de OSRAM Opto Semiconductors. [16]

Fenomenológicamente, algunas diferencias notables entre fotodiodos y fototransistores son las siguientes:

·El fotodiodo es un elemento que genera únicamente una corriente, mientras que el fototransistor genera tanto corriente como voltaje. [17]

·La respuesta del fotodiodo es mucho más rápida que la del fototransistor, y el rango de longitudes de onda para el que puede usarse es mayor, pero a cambio, el fototransistor es mucho más sensible, lo que lo hace mejor candidato a sensor en aplicaciones de colorimetría, como se hará en este proyecto. [17]

Otra característica notable de estos dispositivos optoelectrónicos es su alta directividad: la sensibilidad de los mismos se incrementa drásticamente en función del ángulo que forma la superficie sensible del encapsulado con el haz de luz incidente. En el caso del fototransistor del que se hace uso en este proyecto, la sensibilidad se caracteriza experimentalmente con el ángulo tal y como muestra la siguiente figura, extraída de la quinta hoja de su *datasheet*:

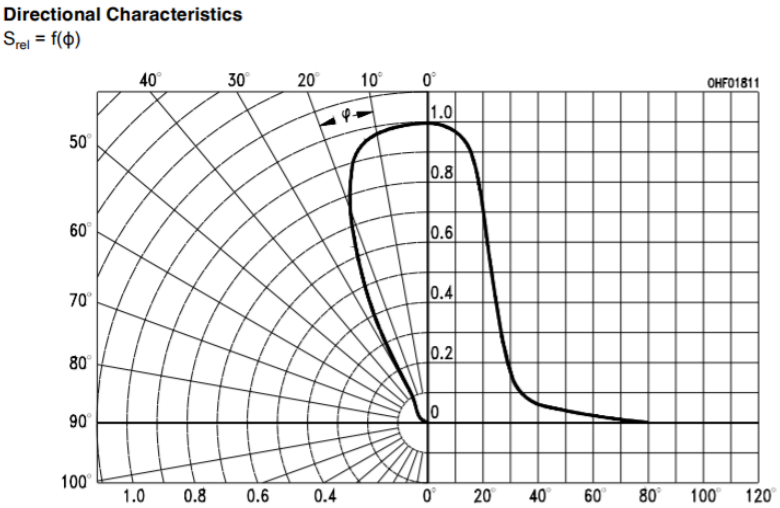

Ilustración 8: sensibilidad relativa del fototransistor en función del ángulo formado por su superficie sensible y su el haz incidente. [18]

El fototransistor se utilizará en el sistema que se implementará para clasificar colores en función de un voltaje. A nivel práctico, su finalidad sería comprobar que el color de un piloto es el apropiado en un determinado instante de tiempo (por ejemplo, algunos teléfonos móviles tienen un piloto en su parte frontal que cambia de color según si se ha recibido una llamada, un mensaje, o si la batería es baja, entre otros parámetros).

2.3-Arquitectura de la tarjeta de desarrollo

El corazón de este proyecto es la tarjeta de desarrollo **Arduino MEGA 2560 Rev3**, que es, a su vez, una versión mejorada y con características más ambiciosas que su predecesora, **Arduino Uno**. Su arquitectura está basada en el microcontrolador **ATmega2560**. Tal y como se describe en el sitio web oficial del fabricante, cuenta con las siguientes características generales: [19]

·54 pines de entrada/salida digital.

·4 puertos serie hardware UART [9].

·Interfaz ISCP [10].

·Conexión USB, y conector de alimentación.

·Oscilador a cristal de 16 MHz.

·Un botón de reset.

Cuenta además con una memoria flash de 256 kB, una SRAM de 8 kB, una EEPROM de 4 kB y tiene además 16 pines destinados a lecturas analógicas de voltaje. La placa cuenta con un conversor analógico-digital de 10 bits, por lo que los resultados de las lecturas analógicas tomarán un valor determinado entre 0 y 1023, incluyendo los extremos. [19]

Dentro de los pines digitales, algunos tienen asignadas ciertas funciones especiales. La tabla que se muestra a continuación muestra resumidamente dichos pines, y las funciones que les corresponden.

Pines en la tarjeta	Función
0, 1, 14, 15, 16, 17, 18 y 19	Comunicación serie, niveles TTL [11]
20,21	Comunicación por protocolo I^2C [12]
2, 18, 19, 20 y 21	Activar interrupciones
50,51,52,53	Comunicación por protocolo MOSI [13]

Tabla 1: funciones adicionales de la tarjeta de desarrollo Arduino MEGA 2560 R3. [19]

Por otra parte, cabe destacar que algunos pines digitales de la tarjeta son controlables a través de registros internos, y que permiten modificar el estado de varios pines digitales simultáneamente.

[9] UART (*Universal Asynchronous Receiver-Transmitter)* es un dispositivo electrónico integrado en computadoras y similares, y que está destinado a la emisión y recepción de datos a través de un solo hilo.

[10] ISCP son siglas para *In-Circuit Serial Programming*. Se trata de una tecnología comúnmente utilizada en sistemas embebidos, y que permite programar dispositivos lógicos tras su instalación.

[11] TTL: *Transistor to Transistor Logic*. Los niveles lógicos son: por debajo o igual a 0.8 V para estado bajo, por encima o igual a 2.2 V para estado alto.

[12] I^2C (*Inter-Integrated Circuit*) es un protocolo para comunicación serie creado por Philips a principios de los 80.

[13] SPI (*Serial Peripheral Interface*) es un estándar desarrollado por Motorola para su familia de controladores 6800, y que cuenta con 4 hilos para las comunicaciones entre controlador y periféricos.

Ilustración 9: fotografía de la tarjeta de desarrollo que se utilizará en el proyecto. [19]

2.4-Programación funcional de la secuencia de test

Se asociará a cada tipo de datos un número de puertos, un rango de valores admisibles, así como uno o dos tipos de ejecución (lectura o lectura/escritura). La siguiente tabla muestra, para cada tipo de dato:

·Los pines de la tarjeta de desarrollo que va a utilizar.

·Los modos (lectura o lectura y escritura) en los que puede ejecutarse.

·Los valores mínimos y máximos tolerables.

·Una información adicional en cada caso.

Tipo	Modo	Puertos	Valores mínimo/máximo	Notas
Voltaje analógico	Lectura/escritura	54 (A0), 55 (A1), 56 (A2), 57(A3)/-	0/5V	El DAC utiliza 8 puertos digitales, pero no son seleccionables.
Sonido	Solo lectura	10,11	0/1	La lectura de sonido es digital.
Temperatura	Solo lectura	60 (A6),61 (A7)	-55ºC/125ºC	Los límites de temperatura son los de la NTC que se usa.
Luminosidad	Solo lectura	64 (A10),65 (A11)	0/5V	La salida proviene de una LDR ubicada en un divisor de tensión.
Frecuencia	Solo lectura	47,49	0.1Hz/1MHz	El rango de frecuencias puede depender del puerto utilizado.
Digital	Lectura/escritura	2,3,4,5/6,7,8,9	0/1	Los valores indicados son niveles lógicos.
Color	Solo lectura	66 (A12), 67 (A13)	-	No existe un rango de valores, sino un valor a comprobar.
Delay	-	-	-	Simplemente genera un retardo en la ejecución del programa.

Tabla 2: estructura funcional del código de etiquetas mediante las cuales se programa el test.

Para todas las instrucciones, los tres primeros caracteres describen el tipo de datos que se maneja, y el cuarto si se trata de una lectura o una escritura (excepto en "DEL"). Con respecto al formato general, a la hora de escribir las instrucciones, pueden seguirse los siguientes pasos, clasificados según el tipo de datos a evaluar/escribir:

·Voltaje analógico ("ANA"), temperatura ("TMP") y luminosidad ("LUZ"): realiza la lectura analógica (ya sea de tensión, temperatura o luminosidad) a través del puerto especificado, y verifica que se encuentra en un rango determinado. En caso de realizar una escritura (únicamente válido para tensión analógica), se utilizará el DAC R-2R para dicho fin. Los valores límite de tensión para tensión analógica y luminosidad se escribirán multiplicados por 10. El valor a escribir, en caso de realizar una escritura analógica se escribe multiplicado por 1000.

·Señales digitales ("DIG") y de sonido ("SND"): sirve para comprobar que se recibe un valor lógico alto o bajo por el puerto que se indique. En el caso de las señales digitales, pueden utilizarse otros puertos para realizarse escrituras. Se escribirá ("DIG") si se espera un valor lógico alto; de lo contrario, se escribirá ("DIG")

·Frecuencia ("FRQ"): verifica que el valor de frecuencia en la lectura se encuentra entre el 95% y el 105% del valor indicado.

·Color ("CLR"): confirma que el color recibido es el mismo que se señala en la instrucción.

De cara al usuario del sistema, y como referencia a la hora de escribir las instrucciones, puede escribirse a modo de resumen, otra tabla que exprese el formato a seguir a la hora de escribir las instrucciones.

Tipo	Puerto	R/W	V1-V2	V
ANA	0,1,2,3/X*	R/W	00-50 (R)	0000-5000 (W)
SND	0,1	R	0000-1111**	-
TMP	0,1	R	00-60	-
LUZ	0,1	R	00-50	-
FRQ	0,1	R	-	1-10000000
DIG	0,1,2,3/0,1,2,3	R/W	-	0000-1111**
CLR	0,1	R	-	COLO***
DEL	-	-	-	0000-9999****

Tabla 3: referencia de las instrucciones necesarias para programar el test. La finalidad de esta tabla es que el usuario del sistema pueda realizar la programación sin grandes complicaciones.

Adicionalmente, podrá introducirse un carácter adicional 'S' que, en caso de encontrarse al final de una instrucción, provocará la parada del test si el resultado de la lectura obtenida en dicha instrucción es erróneo.

Algunas aclaraciones y notas acerca de las instrucciones:

*No significa que haya que escribir explícitamente una 'X', sino que más bien no importa lo que se escriba en su lugar. No obstante, dicha posición no puede quedar vacía (es decir, hay que escribir algún carácter, aunque no condiciona al funcionamiento.

**Puede escribirse “1111”, o simplemente, puede escribirse cualquier otro número distinto de “0000”.

***”COLO” se refiere a las cuatro primeras letras del color que se quiere leer. A la hora escribir la instrucción, sería “BLAN”, “ROJO”, “VERD”, “AZUL”, “AMAR”, “CYAN” o “ROSA”.

****El valor del retardo es teóricamente ilimitado, aunque no tiene sentido hacerlo demasiado largo.

2.5-Estructura del programa general

Veamos ahora como es la estructura del programa general, ubicado en la sección *setup*. Los pasos que se siguen son los siguientes:

1-Conteo inicial del número de pasos a seguir: Se realiza una cuenta de los elementos en el array que contiene las etiquetas descontando todas aquellas que comiencen por `”DEL”`. No afecta a la ejecución del programa, sino más bien a la representación de los resultados a través de la pantalla, ya que se trata de mostrar el número de lecturas y/o escrituras que se van a realizar durante el test.

2-Bucle: Se inicia un bucle *for* que recorre toda posición del array de instrucciones. La estructura interna del bucle puede desglosarse en dos secciones:

2.1-Clasificación inicial: como se mostró en la tabla de códigos, cada tipo de instrucción en el array `inst[]` se asocia con un entero. En esta sección, se actúa consecuentemente dependiendo del valor que dicho entero tome, ejecutando en cada caso determinadas líneas de un *switch* que agrupa todos los casos posibles (8 en total).

2.2-Procesado de los datos: es la zona del bucle que se encuentra dentro del *switch*, y en la que en función del tipo de datos a leer (o escribir), se hace uso de unas determinadas funciones que se explicarán más adelante.

2.3-Resultados: en la tercera sección del bucle, se muestran por pantalla, formateados según el caso, los resultados del test. Además, se hace uso del carácter adicional `'S'`, que sirve para finalizar el test en caso de que la lectura en curso no corresponda con un resultado satisfactorio. Columna por columna, se mostrará el tipo de instrucción en ejecutado en cada caso (por sus tres primeros caracteres), el modo (lectura o escritura), el puerto utilizado, los valores máximo y mínimo tolerados, o el valor esperado, el valor leído, y el tiempo (en milisegundos) necesario para ejecutar la instrucción.

3-Fin del test: simplemente, se muestra el tiempo que se ha requerido para realizar el test y una línea de texto para señalizar que el test ha concluido.

El siguiente diagrama muestra de forma esquemática la estructura del programa:

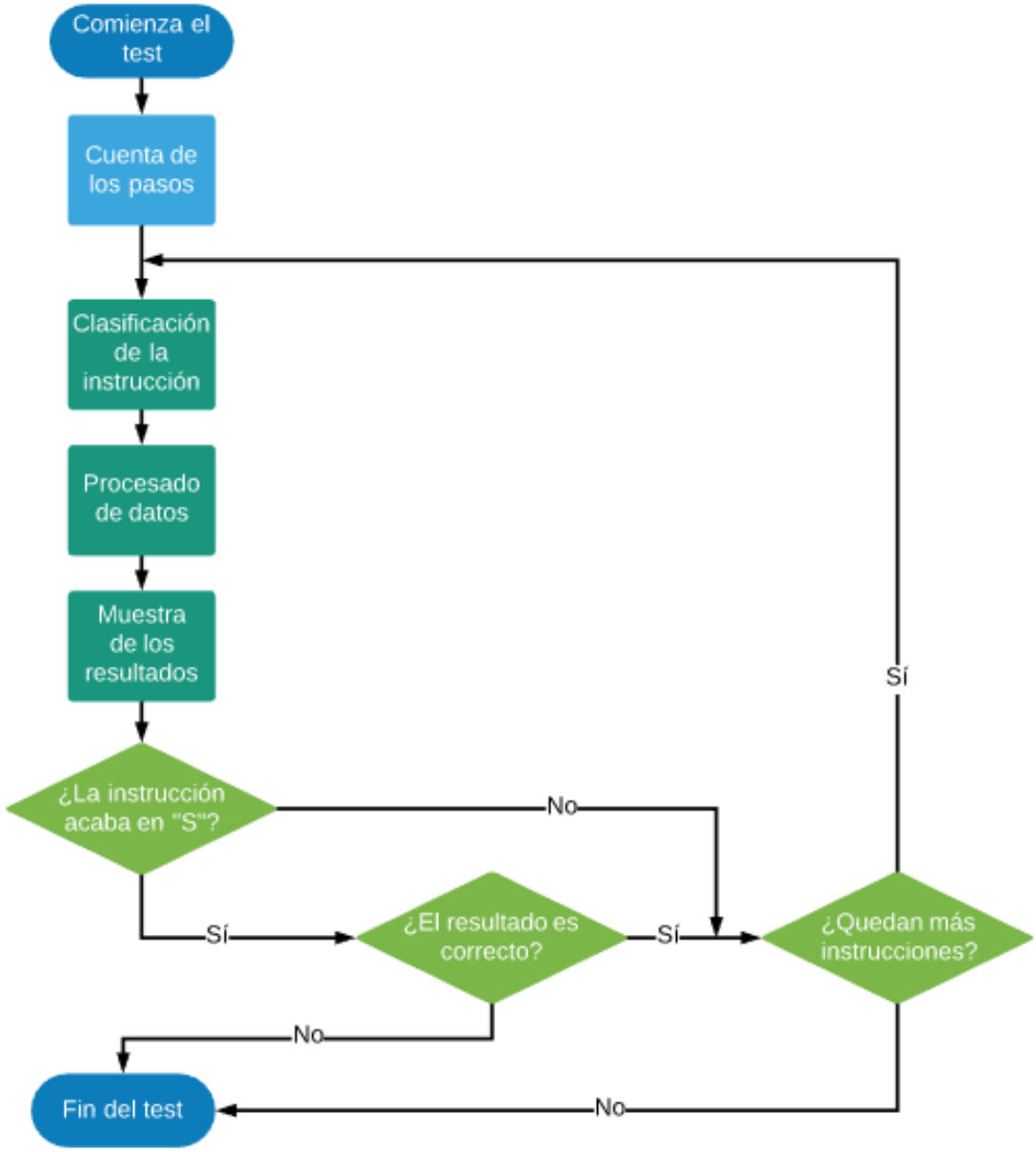

Ilustración 10: diagrama de flujo del programa general.

3-Código en Arduino

El código se escribirá en el compilador de la plataforma **Arduino**. Siguiendo el diagrama de flujo que describe el proceso que ha de seguirse para llevar a cabo las secuencias de test, se almacenarán dichas instrucciones en un *array* de cadenas de caracteres llamado `inst[]`.

Por otra parte, no existen puertos comunes para lecturas/escrituras de distintos tipos de datos. Por ello, es necesario incluir un *offset*, de modo que se pueda pasar de la numeración de puertos de cada modo, a la numeración absoluta de los pines de la tarjeta:

```
#define pRana 54
#define pRdig 2
#define pWdig 6
#define psnd 10
#define ptmp 60
#define pfrq 47
#define pluz 64
#define pclr 66
```

Donde `pRana` es el *offset* para las lecturas analógicas, `pRdig` es el *offset* para lecturas digitales, … Así, si por ejemplo se lee la etiqueta `"ANA2R4550"`, ha de realizarse una lectura/escritura de voltaje analógico, a través del pin **2** asociado a lecturas de este tipo de datos (que corresponde con el pin **56** de la placa), que ha de ser, efectivamente, una lectura, y que ha de encontrarse entre 4.5 y 5 voltios.

Debe facilitarse también, para mejorar la legibilidad del código, así como su programación, una serie de definiciones adicionales, referidas a segmentos de las cadenas (las anteriormente mencionadas etiquetas o instrucciones de la secuencia de test), o bien, conversiones a otro tipo de datos de dichos segmentos:

```
#define tipo inst[j].substring(0,3)
#define pin (inst[j].substring(3,4)).toInt()
#define modo inst[j].charAt(4)
#define volt1 (inst[j].substring(5,7)).toFloat()/10
#define volt2 (inst[j].substring(7,9)).toFloat()/10
#define voltaje (inst[j].substring(5,9)).toFloat()/1000
#define booleano boolean((inst[j].substring(5,9)).toInt())
#define temp1 (inst[j].substring(5,7)).toInt()
#define temp2 (inst[j].substring(7,9)).toInt()
#define freq (inst[j].substring(5,9)).toFloat()
#define color inst[j].substring(5,9)
#define retardo (inst[j].substring(3)).toInt()
```

3.1-Funciones básicas de lectura y escritura

Dentro de todas las funciones que han de escribirse para la correcta lectura y escritura de los datos a gestionar, existen cuatro que resultan de vital importancia. Estas funciones se nombran en el código como *lect_ana, escr_ana, lect_dig y escr_dig*. Además, serán necesarias otras funciones para realizar algunas otras medidas, así como para procesar los datos.

3.1.1-Método de lectura analógica

Es una función muy simple cuyo único argumento de entrada es un entero. La función en cuestión activa como entrada el pin a través del que se lee, y devuelve a continuación el resultado. El valor de la lectura, que se encuentra entre 0 y 1023), ha de ser convertido a un dato

del tipo *float* y multiplicado por la resolución del sistema de lecturas (que incluye al conversor analógico-digital interno de 10 bits), de forma que el resultado sea un valor entre 0 y 5 voltios.

```
float lect_ana(int port){
  pinMode(port,INPUT);
  return float(analogRead(port))*5/1024;
}
```

Se hará uso de esta función cuando tengan que realizarse lecturas de tensión continua, temperatura, y luminosidad.

3.1.2-Método de escritura analógica

Esta función se caracteriza por no tener, a diferencia de todas las demás, un puerto como parámetro de entrada, y más aún, por la dificultad que puede suponer su interpretación, a pesar de su aparente simplicidad.

Es la que hace uso del conversor digital a analógico basado en la red de resistencias R-2R. T-ambién hace uso de la estructura de registros internos de la tarjeta de desarrollo, en particular, del registro **C**, de modo que la escritura en varios puertos digitales puede realizarse simultáneamente. Así que, la función, activa en cualquier caso los 8 bits de los que hace uso el DAC (pines digitales desde el 30 hasta el 37). [20]

El parámetro de entrada es un valor analógico (del tipo *float* en Arduino), y el valor que tome la variable estática declarada en la función (y que será, en este caso un entero), vendrá dada por la siguiente expresión:

$$alta = \frac{2^n}{5} \cdot valor \qquad \textit{(Ec. 8)}$$

Donde el nombre de la variable *alta* hace referencia a los pines que se pondrán en alta en función de los valores que dicha variable tome.

```
void escr_ana(float valor){
  static int alta;
    if(int((pow(2,nbits)/5)*valor)-((pow(2,nbits)/5)*valor)==0&&valor!=0){
      alta=(pow(2,nbits)/5)*valor-1;
    }
    else{
      alta=(pow(2,nbits)/5)*valor;
    }
  DDRC=B11111111;//Activo como salidas todos los pines del 30 al 37,
//extremos incluidos.
  PORTC=alta;//Pone en alta los asociados al intervalo que corresponda.
}
```

Como puede verse en el código, existe una pequeña estructura condicional, que influye a la variable *alta*. El problema está en que el valor que se asocia a la variable es **256** si el parámetro de entrada vale exactamente **5**, por lo que es necesaria la comparación realizada. Así, en caso de que la variable *valor* valga exactamente **5**, la variable *alta* valdrá **255** (11111111_2), de forma

que active todos los pines digitales que se conectan como entradas del DAC. De lo contrario, en lugar de activar todos, simplemente no activaría ninguno.

3.1.3-Método de lectura digital

Se trata de la función cuya ocupación es realizar lecturas digitales. Únicamente toma como parámetro el puerto a través del cual realiza la lectura y retorna el valor leído.

```
boolean lect_dig(int port){
  pinMode(port,INPUT);
  return digitalRead(port);
}
```

Esta función se utilizará para realizar medidas de señales digitales, pero también se utilizará para realizar medidas de señales acústicas, ya que, debido a la naturaleza del test, no interesa saber a qué frecuencia se emite un sonido determinado ni su intensidad, sino más bien saber si el sonido se está produciendo y es detectable.

3.1.4-Método de escritura digital

Esta función es bastante más simple que la correspondiente a la escritura analógica. Simplemente, toma como parámetros de entrada el pin en el que se escribe, así como el valor que se quiere escribir (que será del tipo *boolean*, ya que solo puede ser verdadero o falso). A continuación, activa como salida el pin anteriormente mencionado y escribe el valor lógico en dicho pin.

```
void escr_dig(int port, boolean valor){
  pinMode(port,OUTPUT);
  digitalWrite(port,valor);
}
```

3.1.5-Método de medida de frecuencia

Para realizar medidas de frecuencia, no es suficiente con las funciones explicadas hasta este punto. Se escribirá entonces una función que sea capaz de calcular la frecuencia entrante por un puerto digital, y que hace uso de dos librerías escritas por **Paul Stoffregen**: *FreqMeasure* y *FreqCount*. La primera de ellas es útil para frecuencias relativamente bajas (se recomienda su uso para medidas de frecuencias entre 0.1 Hz y 1 kHz), y basa su funcionamiento en la medida del periodo de la señal. *FreqCount*, en cambio, extiende el rango de medida, aunque trabaja con datos del tipo *unsigned long* en vez de *float*, y basa su funcionamiento en la cuenta de flancos ascendentes para un tiempo fijo. [21] [22]

```
float lect_freq(boolean type){
  if(type==true){
    FreqCount.begin(1000);
    delay(1000);
    if(FreqCount.available()){
      unsigned long frecuencia=FreqCount.read();
      FreqCount.end();
      return frecuencia;
      }
  }
  else{
    float frecuencia=0;
    int veces=0;
    FreqMeasure.begin();
    while(true){
      if(FreqMeasure.available()){
        frecuencia+=FreqMeasure.read();
        veces++;
        if (veces>2000){
          frecuencia = FreqMeasure.countToFrequency(frecuencia/veces);
          FreqMeasure.end();
          return frecuencia;
        }
      }
    }
  }
}
```

A la hora de llamar a la función, se utiliza un parámetro booleano que tiene valor lógico “1” solo cuando el pin que se va a utilizar es el **0** asociado a lecturas de frecuencia (pin digital **47** de la placa). El pin **1** para lecturas de frecuencia es el que se estaría utilizando para medir frecuencias relativamente bajas, y sería el pin digital **49** de la placa, es decir, la librería *FreqMeasure* hace uso del pin digital **49**, mientras que la librería *FreqCount* hace uso del pin **47**. [21] [22]

3.1.6-Método de medida de color

Otra función que es necesario implementar, ya que, a diferencia del resto, no basa su resultado en encontrarse por encima o por debajo de un solo valor, es aquella que se utiliza cuando se quiere medir un color. Su único parámetro es el puerto que se quiere utilizar para la lectura.

```
String lect_color(int port){
  pinMode(port,INPUT);
  String colorLeido;
  float voltajeLeido=float(analogRead(port))*5/1024;
  static int i1;
  for(i1=0;i1<tam;i1++){
    if(voltajeLeido>umbrales[i1]&&voltajeLeido<=umbrales[i1+1]){
      colorLeido=colores[i1];
    }
  }
  return colorLeido;
}
```

El código hace uso de algunas variables que todavía no se han explicado:

·*colorLeido* es una cadena de caracteres que a su vez representará el resultado de la función. Esta función devolverá el nombre del color que se ha leído.

·Para ello, se hará uso de un *array* de números reales llamado *umbrales*, y que contiene los valores de tensión que delimitan los rangos de voltaje con los que se asocia cada color.

·La tensión medida y que se comparará con los valores del *array* llamado *umbrales* se almacena en una variable real llamada *voltajeLeido*.

·*i1* es un número entero que se utiliza simplemente para iterar.

3.1.7-Método de validación de resultados

Además, necesitaremos también una función adicional general a las lecturas de todo tipo de datos que se maneja, excepto para color y señales digitales. Las lecturas de sonido, voltaje analógico, temperatura, luminosidad y frecuencia, pueden agruparse por la validez de los resultados: las señales medidas se considerarán correctas si se encuentran en un rango previamente especificado. Por ese motivo, puede escribirse una función utilizada para este fin:

```
boolean compara(float valor, float minimo, float maximo){
  return valor>=minimo&&valor<=maximo;
}
```

Y que simplemente devuelve un dato del tipo *boolean*, *true* si el resultado es correcto, y *false* si no lo es. Cabe destacar que en algunos casos especiales (medidas de color, sonido y señales digitales), no puede usarse esta función, sino que el valor booleano que indica la validez del resultado se obtiene comprobando si el valor lógico o el color leído es igual al esperado o no.

3.2-Estructura interna del switch

Tal y como se comentó al explicar la estructura del programa general, dentro de la estructura del tipo *switch* han de tomarse algunas decisiones para hacer uso de las funciones básicas de lectura y escritura (así como de otras funciones auxiliares). Cabe destacar que, siempre y cuando la instrucción escrita no sea de retardo (`"DEL"`), se guardará el puerto que se quiere usar en una variable llamada *puerto*, y el modo (lectura o escritura) en una variable llamada *modo*. Para discriminar, cada caso, en función del tipo de dato que se quiera manejar, se usa la función *enumera*, que asocia un número con cada tipo:

```
int enumera(String subc){
  if(subc=="ANA"){
    return 0;
  }
  else if(subc=="SND"){
    return 1;
  }
  else if(subc=="TMP"){
    return 2;
  }
  else if(subc=="LUZ"){
    return 3;
  }
    else if(subc=="FRQ"){
    return 4;
  }
  else if(subc=="DIG"){
    return 5;
  }
  else if(subc=="CLR"){
    return 6;
  }
  else if(subc=="DEL"){
    return 7;
  }
}
```

La variable *puerto* dará la numeración absoluta del pin de la tarjeta de desarrollo que se utilizará; por ello, se obtendrá su valor sumando el *offset* correspondiente al tipo de dato al número del pin que se quiere usar de cada tipo de dato. Por ejemplo, si se usa el pin **1** de lecturas de temperatura, *puerto=ptmp+1*=60+1=**A7**. Adicionalmente, y si se quiere realizar una lectura de cualquier tipo, se utilizará una variable de tipo *float* llamada *lectura*, que luego se comparará con los valores especificados para validar el resultado.

Excepto para el caso **7** del *switch* (es decir, la instrucción de espera), el diagrama de flujo a seguir para la clasificación y procesado de las instrucciones es el siguiente:

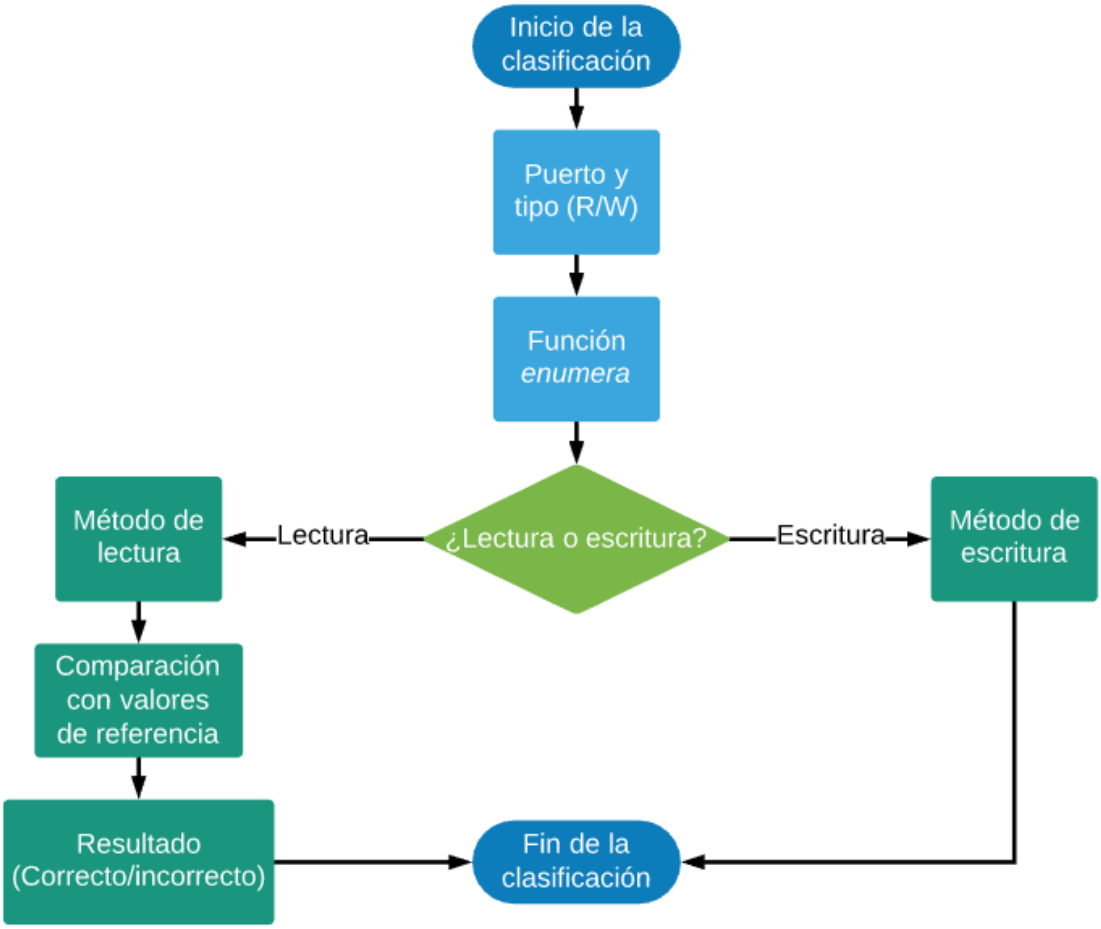

Ilustración 11: diagrama de flujo para la clasificación y procesado de las instrucciones.

4-Implementación del sistema de test

Además de escribir el programa en el IDE[14] de Arduino, el proyecto consta de una parte de desarrollo de hardware. Se procede entonces, a montar un circuito completo que incluya todos los sensores y actuadores necesarios, tanto para realizar las medidas necesarias, como para generar las señales entrantes al sistema bajo pruebas.

Para ello, se construye un sistema conformado por el mencionado circuito y dos tarjetas de desarrollo Arduino MEGA 2560 R3. En lo que sigue, se nombrará la tarjeta que realiza las medidas y gestiones necesarias como *tarjeta de medida*.

Veremos, a continuación, parte por parte, como se ha implementado cada circuito necesario para realizar las medidas. El código de colores será el siguiente:

·Naranja: salidas digitales.

·Amarillo: entradas digitales.

·Verde: entradas analógicas.

·Morado: salidas analógicas.

·Negro: tierra.

·Rojo: alimentación de 5 V.

·Blanco: alimentación de tensión superior a 5 V.

·Azul: conexiones entre componentes de un mismo circuito.

·Gris: señales PWM.

4.1-Conversor digital-analógico

Tal y como se explicó en el marco teórico de este proyecto, el DAC es de especial utilidad a la hora de generar señales analógicas cuando las tarjetas de desarrollo utilizadas utilizan métodos que pueden ofrecer resultados similares, pero pudiendo dañar, según el contexto, los dispositivos a testear. La alternativa más sencilla, como se mencionó anteriormente, es construir un DAC basado en una red de resistencias (en este caso, de 1 kΩ y 2 kΩ) y un seguidor de tensión. Las entradas del conversor irían conectadas a los pines digitales del registro **C** de la tarjeta de medida, es decir, aquellos desde el **30** hasta el **37**. [20]

[14] IDE son siglas para *Integrated Development Environment*, es decir, una herramienta informática destinada a la asistencia para el desarrollo de software.

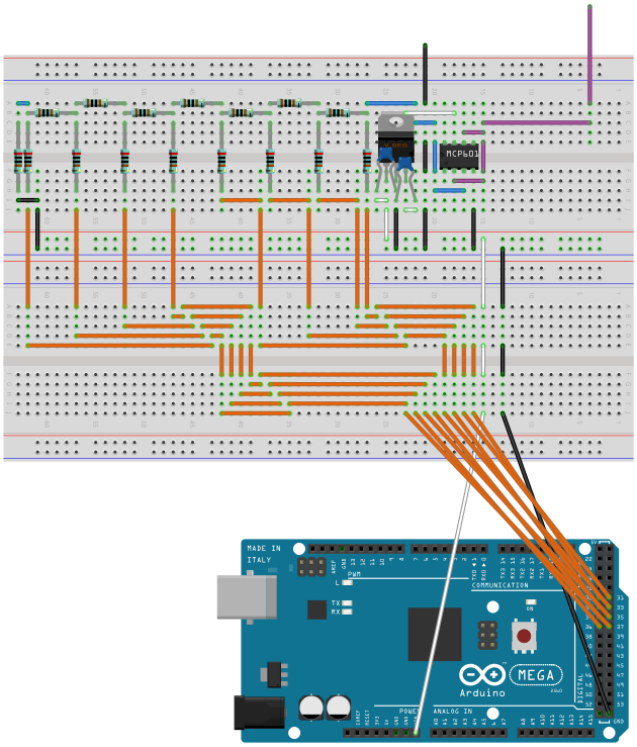

Ilustración 12: esquema del montaje del DAC R-2R.

Para implementar el seguidor de tensión a la salida de la red de resistencias, se utiliza un amplificador operacional. Debido a que la tarjeta no puede entregar a su salida voltajes negativos, lo correcto es utilizar un amplificador que precise únicamente de una alimentación positiva. El modelo de OPAMP utilizado es el MCP601 de Microchip. Por otra parte, es necesario alimentar dicho amplificador con una tensión de 6 voltios [23]. Para poner a prueba el sistema, puede utilizarse el siguiente programa:

```
#define nbits 8
float f;

void setup() {
  Serial.begin(57600);
  pinMode(A0,INPUT);
}

void loop() {
  escr_ana(f);
  f+=0.01;
  Serial.println(float(analogRead(A0))*5/1024);
}

void escr_ana(float valor){
  static int alta;
  if(int((pow(2,nbits)/5)*valor)-((pow(2,nbits)/5)*valor)==0&&valor!=0){
    alta=(pow(2,nbits)/5)*valor-1;
  }
  else{
    alta=(pow(2,nbits)/5)*valor;
  }
  DDRC=B11111111;
  PORTC=alta;
}
```

El objetivo del programa es generar ondas de *diente de sierra*, utilizando para ello la función de escritura analógica, e incrementando su argumento en 0.001 (mediante la variable *f* de tipo *float*) en cada iteración de un ciclo sin condición de parada. Sin embargo, el buffer a la salida del DAC no puede ser alimentado con 5 voltios, ya que se obtiene un resultado erróneo.

En cambio, el resultado es el esperado si la tensión de alimentación pasa a ser de 6 voltios. Para ello, ha de utilizarse el pin **Vin** en la tarjeta de desarrollo. Este pin entrega un voltaje igual a aquel con el que se alimenta la tarjeta. Teniendo en cuenta que vamos a utilizar una fuente de alimentación de 9 V, puede usarse dicho pin junto con un regulador de tensión de 6 voltios (en este caso, el modelo **MC7806CTG** de **On semiconductors**). Los resultados para esta segunda implementación son satisfactorios:

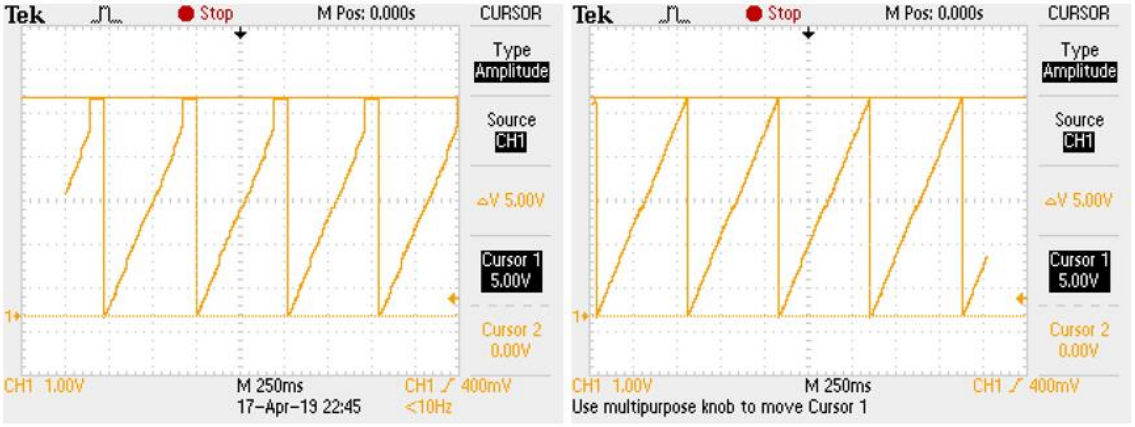

Ilustración 13: formas de onda generada por el DAC, para un voltaje de alimentación positivo del OPAMP de 5 voltios (izquierda) y una tensión de alimentación positiva de 6 voltios (derecha).

Para caracterizar, finalmente, el DAC por su error absoluto, se escribirá otro programa. Igual que en el anterior, la tensión se aumenta en 1 mV, aunque esta vez el número de iteraciones se limita a 5001. El resultado será el valor de *f* para el cual el error es máximo, junto con dicho error:

```
void setup() {
  Serial.begin(9600);
  pinMode(A1,INPUT);
  escr_ana(0);
  for(i=0;i<=5000;i++){
    escr_ana(f);
    delay(50);
    error=abs(f-(float(analogRead(A1)))*5/1024);
    delay(50);
    Serial.println(error);
    Serial.println(i);
    if(error>err_max){
      err_max=error;
      f_err_max=f;
    }
    f+=0.001;
  }
  Serial.println("Error máximo: (f_err_max,
err_max)=("+String(f_err_max)+", "+String(err_max)+")");
}
```

Donde **f**, **error**, **err_max** y **f_err_max** son variables del tipo *float* declaradas previamente.

Finalmente, se obtiene que el error máximo es de **0.09** voltios cuando el voltaje esperado a la salida es de **2.69** voltios.

4.2-Medidores de temperatura

Para implementar un medidor de temperatura, la opción más sencilla consiste en utilizar una NTC. Concretamente, va a utilizarse la NTC **B57164K0103J000** de **EPCOS**, cuyo coeficiente de temperatura es de 4300 ºK y su resistencia a 25 ºC es de 10 kΩ (con lo que el valor de β, y el de la resistencia a la temperatura de referencia, R_0 quedan ya completamente definidos en el programa). [24]

También se instalará, como se explicó en el diseño a alto nivel, una NTC para la cual se conocen los coeficientes de **Steinhart-Hart**.

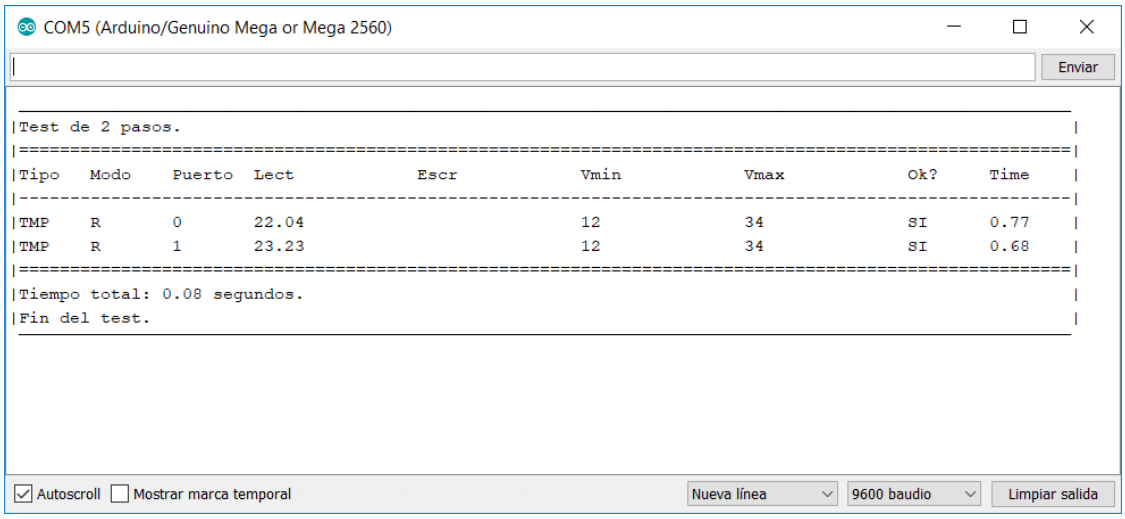

Ilustración 14: prueba comparativa de medida de la temperatura usando ambos termistores. La diferencia es algo superior a un grado centígrado.

Para probar la fiabilidad de los termistores, puede compararse la medida realizada con otra, a través de un termómetro basado en un termopar. Como se observa en la siguiente imagen, el resultado obtenido es similar con respecto a la NTC ubicada en el puerto **0**, aunque con mayor diferencia con respecto a la utilizada en el puerto **1** debido al autocalentamiento de la misma:

Ilustración 15: medida de temperatura realizada en el laboratorio de electrónica, en iguales condiciones que el sistema de test, utilizando un medidor alternativo.

4.3-Sensores ópticos

En este apartado, se muestra el uso que se le da en el presente proyecto a elementos sensibles a la luz. Por una parte, se utiliza una resistencia dependiente de luz (LDR), que podría utilizarse también para medir color, aunque no es la mejor opción, y por otra parte se utiliza un fototransistor, que ofrece una respuesta mucho más rápida que la LDR para medidas de color, así como una mayor directividad.

4.3.1-Sensor de luminosidad

Para medir luminosidad, vamos a utilizar una LDR del fabricante Luna Optoelectronics (modelo NSL-19M51). Esta fotorresistencia cuenta con una resistencia de oscuridad de 20 MΩ, una resistencia mínima a la luz de 20 kΩ, y una máxima de 100kΩ. [25]

Puesto que la fotorresistencia ofrece una mayor resistencia cuanto menor es la iluminación que incide sobre la misma, lo más adecuado es colocarla dentro de un divisor de tensión, entre la tensión de alimentación (5V) y el nodo de salida de dicho divisor. De esta forma, el voltaje a su salida será creciente con la iluminación incidente sobre el elemento sensor. Si se utiliza como carga una resistencia de 10 kΩ, se logra una salida prácticamente nula en oscuridad, y unos 4.5 V a su salida al ser iluminado.

4.3.2-Sensor de color

Como sensor de color, puede ser suficiente un fototransistor conectado con su colector a la alimentación, su emisor a la salida junto con la resistencia de carga, es decir, en colector común (ya que la base no puede conectarse, y su corriente se genera por iluminación). En este caso, la resistencia conectada entre emisor (salida del circuito sensor de color) y la referencia a tierra es de 1 MΩ. Es necesario que dicha resistencia sea grande, para que las corrientes del fototransistor puedan dar lugar a un voltaje razonable. Según se indica en la segunda hoja de su datasheet, su corriente de colector máxima es de 50 miliamperios. [18]

También se muestra, en la misma hoja de características, una gráfica que indica la sensibilidad relativa del dispositivo en función de la longitud de onda de la luz incidente, y que se muestra en la siguiente figura:

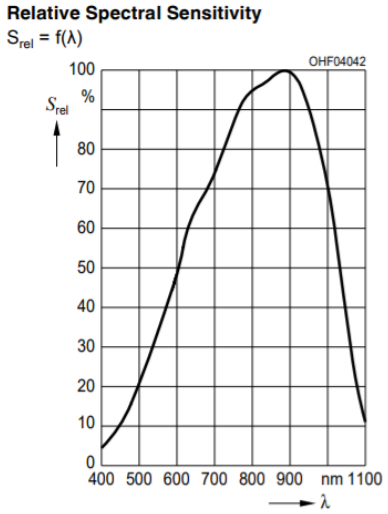

Ilustración 16: gráfica que muestra como varía la sensibilidad relativa del fototransistor SFH 310-2/3 en función de la longitud de onda de la luz incidente. [18]

Aun así, existiría un inconveniente, y es que las condiciones lumínicas del entorno pueden afectar dramáticamente a los voltajes asociados a cada color, pudiendo incluso cambiar la posición relativa de voltajes y colores en la lista. Por ejemplo, puede ocurrir que el color azul se asocie con un voltaje menor que el verde en la mayoría de los casos, aunque en determinadas circunstancias no ocurre así. En apartados siguientes, se explicará la calibración del sensor, junto con la solución al recién mencionado problema de la ordenación de voltajes y colores.

4.4-Micrófono

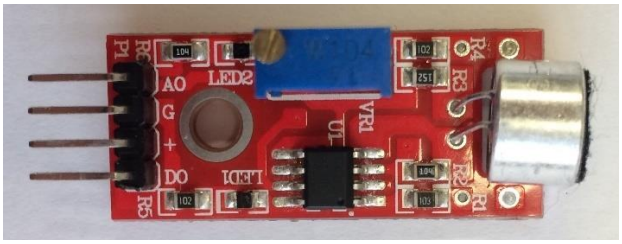

Ilustración 17: fotografía del micrófono utilizado, realizada en el laboratorio.

Para ello, el sistema de test planteado incluye un micrófono *electret* preamplificado (modelo KY-038, fabricado por **Keyes**), montado sobre un PCB, que cuenta además con un ajuste manual. Debido al enfoque funcional del test, no interesa medir la magnitud del sonido producido, sino más bien, si el sonido es audible o no, es decir, ha de funcionar como un sensor "todo o nada". El objetivo del ajuste (en color azul en la figura 20) es ajustar el umbral a superar para que la señal acústica medida dé lugar a un "1" lógico.

5-Implementación del sistema bajo pruebas

Además de un sistema de medida, se implementará un circuito sencillo con el que ponerlo a prueba. Dicho circuito generará señales del mismo tipo que el sistema de test implementado ha de medir. También recibirá señales del sistema de test, que tendrán que ser mostradas para verificar su correcto funcionamiento. Contará para ello con los componentes que se explican a continuación, y su arquitectura responderá al siguiente diagrama de bloques:

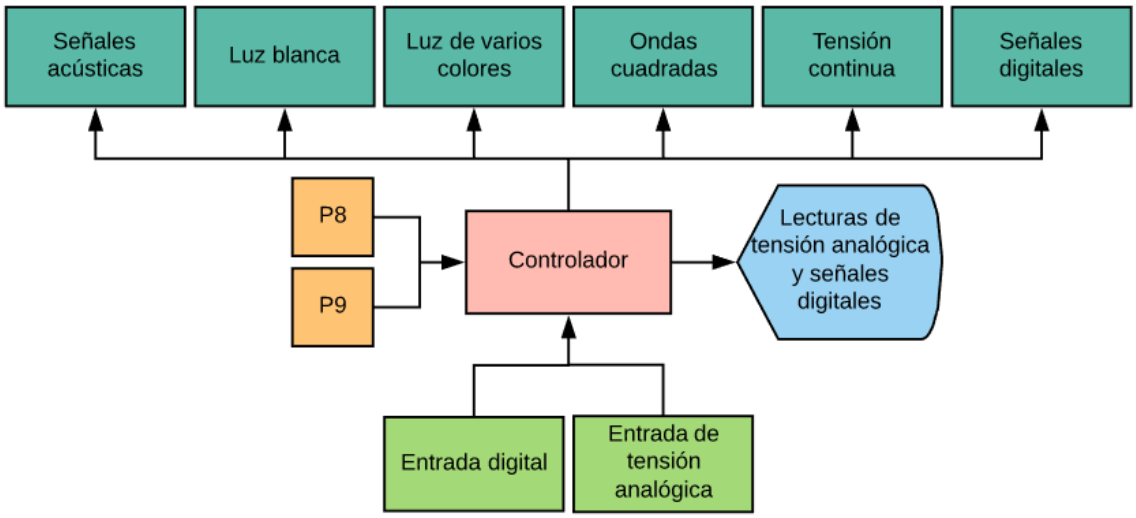

Ilustración 18: diagrama de bloques del sistema bajo pruebas.

Donde **P8** y **P9** son señales digitales provenientes del sistema de test, como se explicará más adelante.

5.1-Buzzer

Un zumbador o *buzzer*, es un dispositivo de dos terminales que contiene en su interior una fina lámina de un material piezoeléctrico, es decir, que cambia su forma según sea la diferencia de potencial eléctrico entre dos de sus extremos. Al cambiar su forma, emite sonidos en frecuencias audibles, y normalmente puede ser controlado con señales PWM (como es el caso). [26]

Para el montaje de este simple circuito, tan solo hay que conectar uno de sus terminales (marcado como positivo) a una salida digital PWM de la tarjeta de test, y el otro terminal a la referencia a tierra.

5.2-Emisores de luz

Se explica a continuación el montaje de dos circuitos muy simples que formarían parte de la "circuitería de test": el LED blanco y el LED RGB.

5.2.1-LED blanco

Se trata de un diodo emisor de luz que se conectará en serie con una resistencia de un valor relativamente bajo (220 Ω), para que la luz emitida sea bastante fuerte.

5.2.2-LED RGB

Es una modificación del LED habitual, que consta de 4 terminales. Uno de ellos conecta el diodo a tierra, mientras que los otros tres se corresponden cada uno con un color. Se utilizan también resistencias de 220 Ω que unen las conexiones digitales del *Arduino de test* con el LED. Así, diferentes valores de las salidas digitales de dichos pines digitales darán lugar a distintos colores, medibles con el sensor de color.

5.3-Potenciómetro

La finalidad de esta parte del circuito es la de entregar a su salida una tensión variable manualmente, que sea después detectable por la tarjeta de medida.

Es también un subcircuito muy simple, ya que únicamente se compone por una resistencia en serie (de 100 Ω) con un potenciómetro. El uso de la resistencia en serie se debe a que el modelo de resistencia variable utilizado tiene una resistencia mínima muy baja, con lo que puede llegar a quemarse si se somete a ciertas tensiones (que no tienen por qué ser excesivamente altas).

5.4-Oscilador

El principal objetivo de este circuito es, como su nombre sugiere, generar señales de frecuencias seleccionables y ajustables, en este caso, por resistencias conocidas.Se utilizará para ello, la configuración *A-stable* del chip NE555[15]. La figura presente en la pág. 11 de su *datasheet*, explica como han de realizarse las conexiones para este modo de operación: [27]

En estas circunstancias, el tiempo en alta del oscilador con respecto al tiempo en baja (es decir, su *duty cycle*), y la amplitud pueden ser relevantes. Por ello, se incluye un potenciómetro en la salida de dicho circuito oscilador que regule el nivel de salida, ya que, a altas frecuencias, se producen transitorios que superan el nivel de tensión esperado.

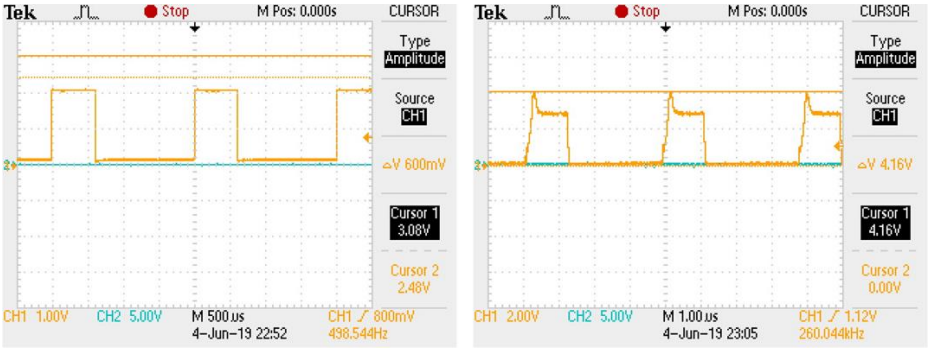

Ilustración 19: formas de onda a mínima (izquierda) y máxima frecuencia (derecha) alcanzables por el oscilador construido.

Así, el "circuito actuador", junto con el circuito que contiene el DAC y todos los circuitos de medida necesarios, quedaría como se muestra en la siguiente imagen:

[15] El comúnmente conocido como "555", es un circuito temporizador creado por **Hans Camenzind** en 1971, e inicialmente comercializado por **Signetics**. Debido a su versatilidad y bajo precio (75 céntimos de dólar en su primer lanzamiento al mercado), es uno de los circuitos integrados más populares de la historia de la electrónica. [30]

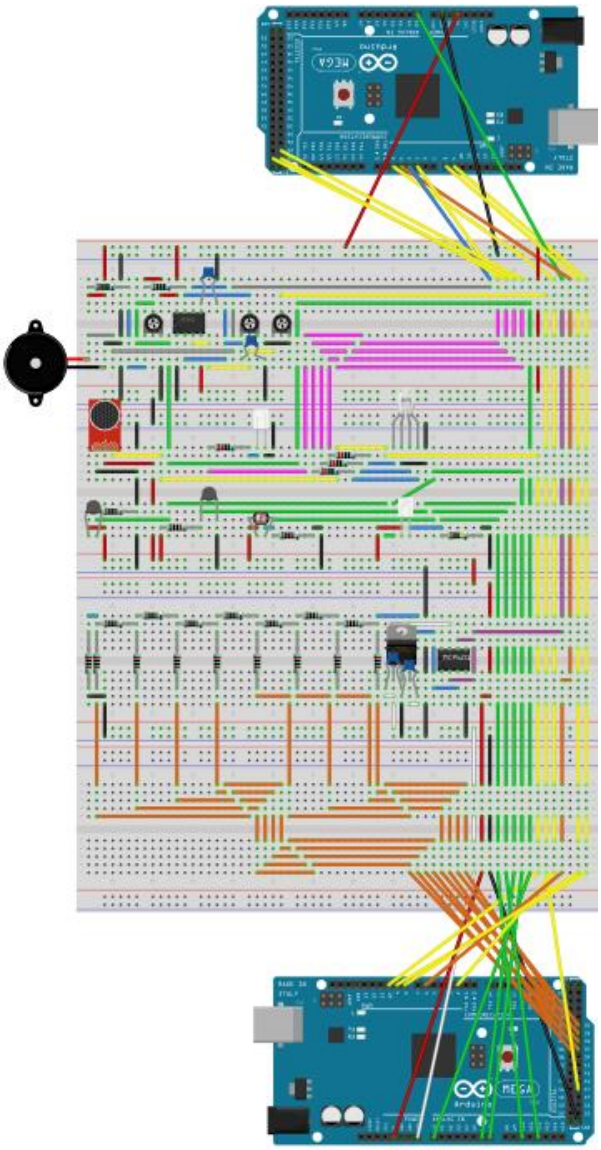

Ilustración 20: esquema del circuito que contiene todos los subcircuitos mostrados y explicados en los anteriores apartados, junto con el cableado necesario para comunicar ambas tarjetas.

Adicionalmente, se comunicarán ambas tarjetas a través de dos señales digitales, emitidas por la tarjeta de medida y recibidas por la tarjeta de test, que controla el *Device Under Test*. Esta tarjeta de test activará los LEDs o el *buzzer* cuando sea necesario. También mostrará por pantalla el valor de una lectura analógica y otra digital. Así, podrán ponerse a prueba los sensores de luz y color y el micrófono, pero también podrá comprobarse que el DAC emite un voltaje correcto y que la señal digital escrita por el sistema de test es correcta.

Para llevar a cabo la comunicación, se utilizarán los pines **38** y **39** de la tarjeta del sistema de test como salidas, y los pines **8** y **9** (respectivamente), como entradas. Los bits correspondientes se llamarán *P8* y *P9*. La siguiente tabla explica la función que ha de realizar el Arduino del sistema bajo pruebas en función de las señales digitales recibidas:

P8	P9	Acción
0	0	Se enciende el LED blanco
0	1	Se activa el *buzzer*
1	0	Se muestra por pantalla la lectura digital y la analógica
1	1	Se activa el LED RGB

Tabla 4: función lógica que ejecuta la tarjeta de test, y que establece la comunicación con la tarjeta de medida.

6-Mejoras del sistema de test

El circuito, tal y como se ha mostrado hasta ahora, cumple con las funciones básicas para llevar a cabo el test electrónico. Sin embargo, puede añadirse una secuencia adicional en el código, que mejore la calibración del sensor de color, y un método de almacenamiento de datos en una memoria externa, como se explicará a continuación.

6.1-Memoria externa

Hasta este punto, el sistema es capaz de realizar secuencias de test y mostrar por pantalla los resultados del mismo, pero surgen dos limitaciones relevantes:

·El sistema ha de estar conectado a un ordenador para monitorizar los resultados.

·El programa ha de modificarse al cambiar la secuencia de test, ya que ha de cambiarse el contenido dentro de un *array*.

Una solución efectiva para ambos problemas pasa por ampliar el sistema de test con un módulo de lectura/escritura de memorias externas. De esta forma, la secuencia de test podría programarse en un archivo de texto en dicha memoria, pero también podrían almacenarse en la misma los resultados de los test realizados.

Para ello, se instalará un módulo para tarjetas microSD[16] de la marca **kwmobile**, montado sobre el circuito y conectado a la tarjeta de medida. También se modificará el programa cargado en la tarjeta de medida, para que actúe tal y como se muestra en el siguiente diagrama:

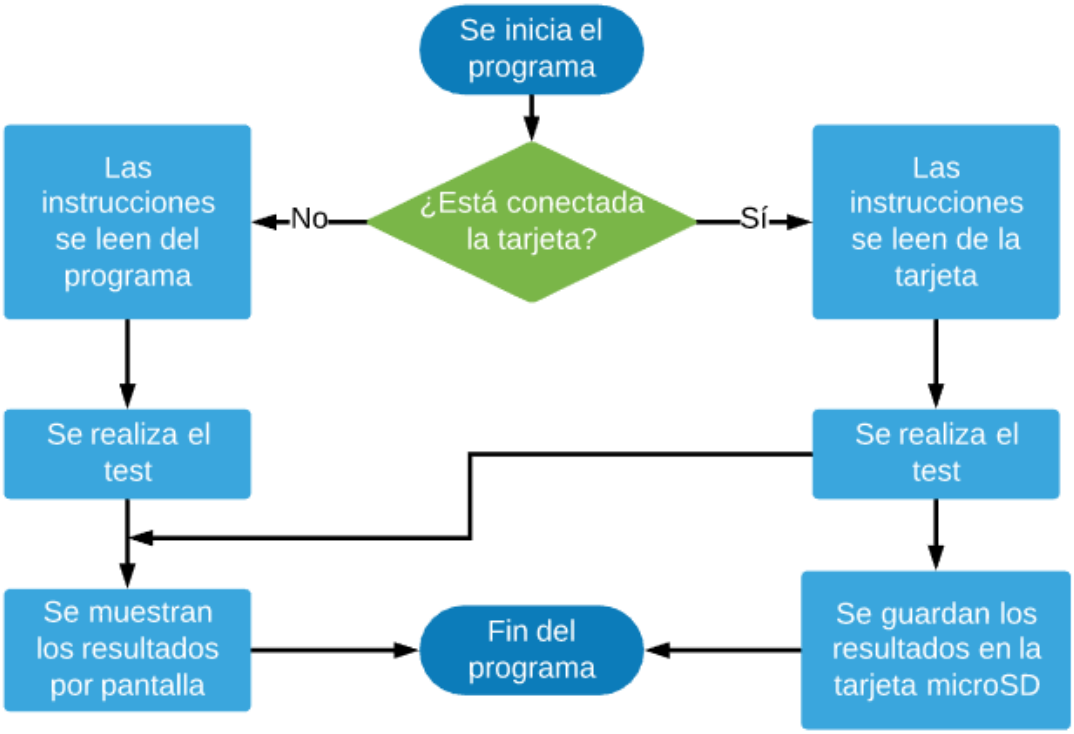

Ilustración 21: diagrama de flujo que muestra cómo se ejecuta el test si la tarjeta se encuentra conectada o no.

[16] SD (*Secure Digital*) es un estándar de tarjetas de memoria comúnmente adoptado en la actualidad. Su tamaño más común (microSD) es, hoy por hoy, el formato preferido por muchos de los fabricantes de teléfonos móviles.

La programación de las instrucciones se verá facilitada ya que tan solo es necesario escribir una instrucción por línea en un archivo llamado "*test.txt"*. Los resultados se escribirán en una tarjeta llamada *"res"*, y se escribirán en un archivo de texto llamado "*pxx.txt"*, donde *"xx*" hace referencia al número del test.

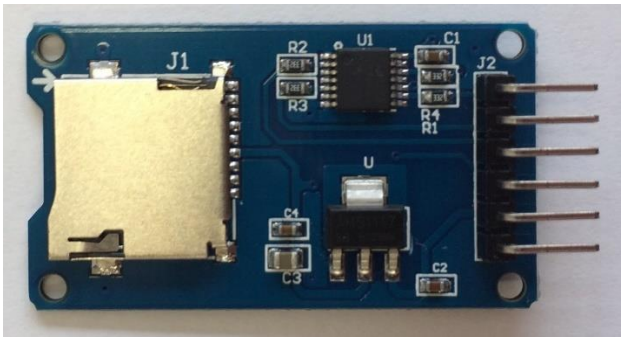

Ilustración 22: fotografía adquirida en el laboratorio del módulo lector de tarjetas microSD.

6.2-Secuencia de autocalibración del sensor de color

Antes de utilizar el sensor de color, es necesario realizar una calibración del sistema. Para ello, se tendría que emitir cada color a través del LED RGB, anotar el valor de voltaje a la salida del sensor en cada caso, y definir después varios rangos de tensión para posteriormente ser asociados con cada color. Sin embargo, los cambios en las condiciones del entorno pueden afectar a las medidas. Para facilitar las calibraciones, implementarse un sistema que realice automáticamente los siguientes pasos:

·Crear un arreglo[17] que contenga en cada posición una cadena de caracteres con la que se identifique cada color, y un *array* vacío del tipo *float* de igual longitud.

·Emitir, en el mismo orden que en el arreglo de ***Strings***, cada color a través del diodo LED de varios colores, y guardar en memoria cada medida de voltaje realizada por el sensor de color.

·Utilizar un algoritmo de ordenación para ordenar cada color por su voltaje asociado, de menor a mayor. Nótese que un cambio en el *array* en el que se guardan los voltajes leídos también ocasiona un cambio en el de cadenas de caracteres.

·Generar un último arreglo de números de coma flotante (*float*) llamado *umbrales*, en el que se guarden los rangos asociados a cada color. Para ello, se hará la media aritmética de cada par de voltajes del *array* anterior. Para los voltajes mínimo y máximo del *array* creado en el paso anterior, se hará la media con 0 y 5 V.

Como resultado, se obtiene una lista en la que se indica el voltaje al realizar la lectura para cada color, así como el contenido del *array* que se usará como referencia para detectar los colores.

[17] En ocasiones, se utiliza la palabra *arreglo* como sinónimo de *array*.

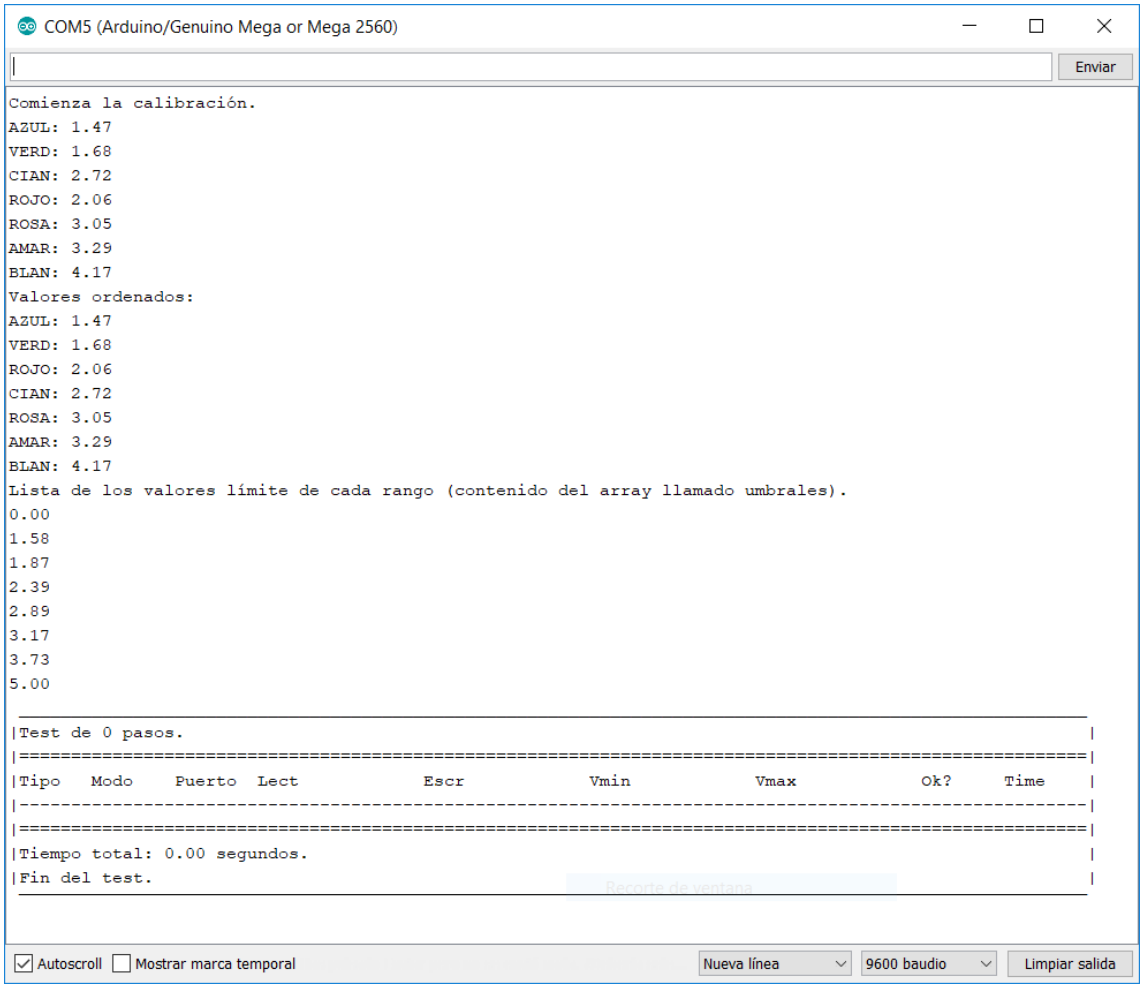

Ilustración 23: resultado mostrado por pantalla a través de la tarjeta de medida, al introducir la instrucción "CAL".

Se hace uso de una función llamada *calibraColor*, con tres parámetros de entrada:

·Un arreglo de números reales.

·Otro arreglo de cadenas de caracteres que incluya los colores que se quieren evaluar.

·El número de colores.

·El puerto que se utiliza para medir color en el sistema (**0** para medir a través del pin **A12**).

Esta función hace uso del algoritmo de ordenación *Bubble Sort*, adaptado de una implementación en **Arduino** extraída de la página **Hardware Hacks** [28].

Con respecto al hardware, será necesario conectar los pines digitales **22**, **23** y **24** de la tarjeta de medida al LED RGB, de forma que cada pin se corresponda, respectivamente, con los colores azul, verde y rojo. Serán manipulados a través del registro **A** de la tarjeta de medida.

7-Prueba de funcionamiento

En este último apartado del desarrollo práctico del proyecto, se explicarán algunas correcciones necesarias para poder llevar a cabo exitosamente el test, así como los resultados extraídos de un test que ponga a prueba la fiabilidad del sistema.

7.1-Correcciones del hardware previas a la prueba

Por una parte, ha habido que cambiar el cableado de la alimentación y tierra de los sensores. Con el cableado tal y como se muestra en el diagrama de todo el circuito, la tensión puede verse reducida en, aproximadamente, 0.5 voltios tras el paso por varias secciones horizontales de las *protoboard*.

Por otro lado, las lecturas de frecuencia no son óptimas si el ciclo de trabajo de las señales medidas no es del 50%. Para ello, se sustituye el oscilador basado en el chip **NE555** (desconectándolo de la alimentación) por un generador de ondas cuadradas basado en el circuito integrado **HEF4046BP** de **Philips Semiconductors**.

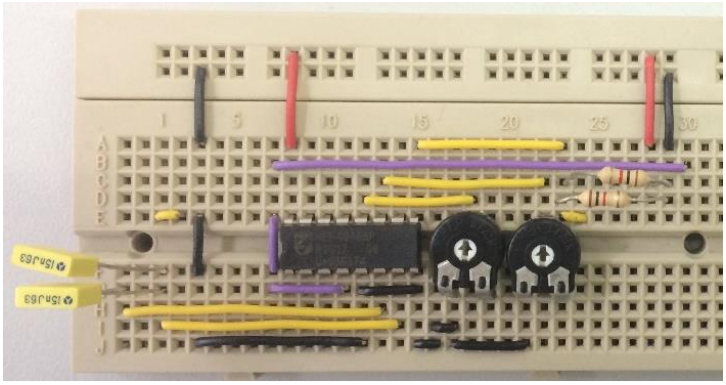

Ilustración 24: generador de ondas cuadradas con ciclo de trabajo del 50%, basado en el circuito HEF4046BP de Philips Semiconductors.

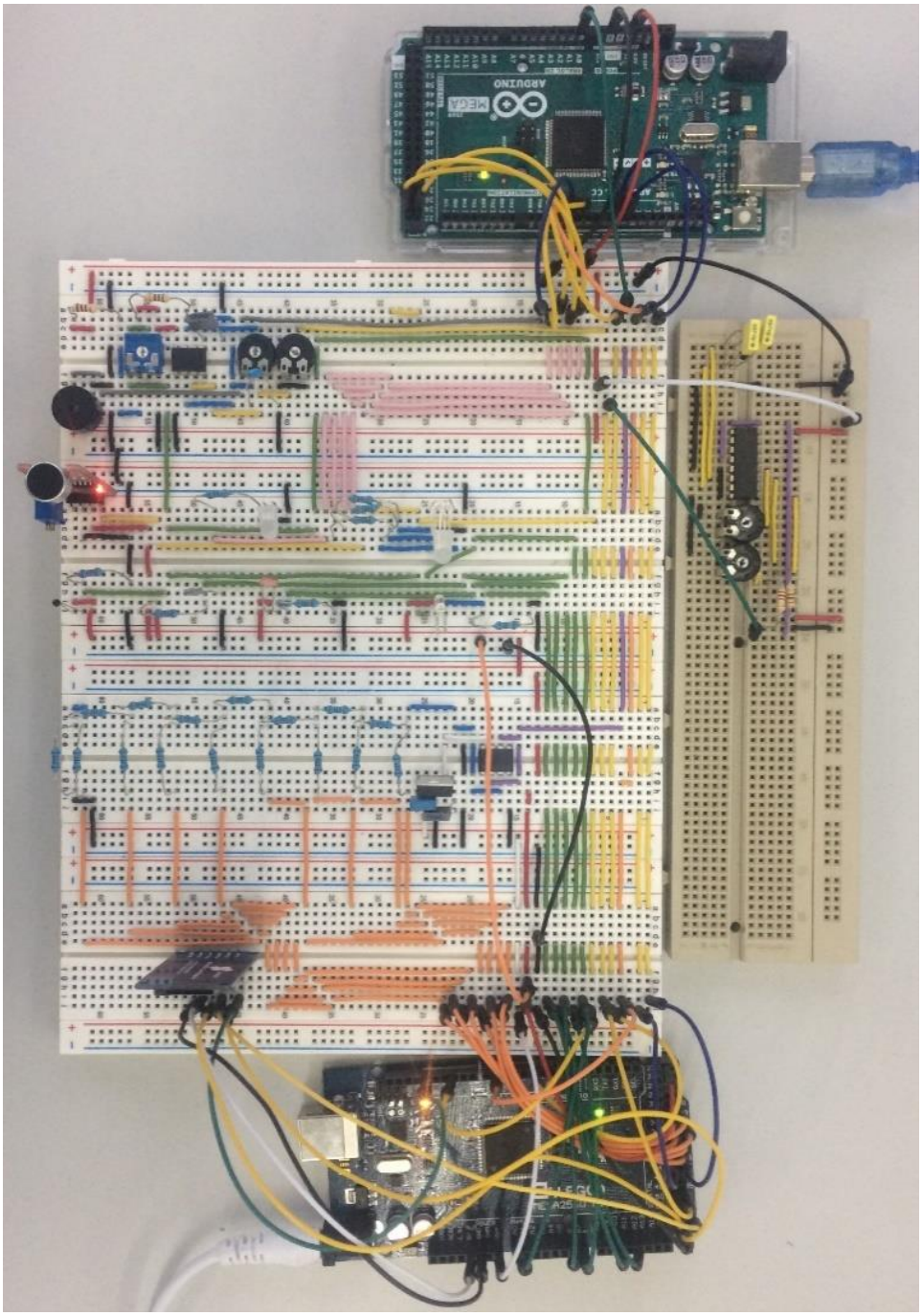

Ilustración 25: imagen real del circuito sobre el que se implementa el sistema de test.

7.2-Test final

Para poner a prueba el sistema, se llevará a cabo una secuencia que involucre todos los sensores del sistema, y con la que se realicen todas las medidas posibles. Con ese fin, se escribirá en el programa correspondiente a la tarjeta de medida un programa con todas las instrucciones posibles (es decir, instrucciones mediante las que se midan todas las magnitudes físicas para las que el sistema se ha diseñado, y se realicen escrituras tanto analógicas como digitales):

```
"ANA0W0000","ANA0R2022","ANA0W4500","TMP0R2030","TMP1R2030","LUZ0R4050
","FRQ0R1000","DIG0R0000","DIG0W1111","CLR0RROJO","SND0R1111"
```

Por otra parte, es necesario también generar algunas magnitudes para realizar la prueba:

·Un voltaje determinado y conocido.

·Una señal de sonido, activando el *buzzer*.

·Luz blanca, utilizando el correspondiente LED.

·Una onda cuadrada de 1 kHz.

·Una señal digital en nivel lógico alto.

·Luz roja, mediante la activación del LED RGB.

La onda cuadrada se generará, por los motivos anteriormente razonados, utilizando el VCO contenido en el lazo de enganche de fase, y el voltaje se regulará con el potenciómetro incluido en el circuito, y medido con uno de los voltímetros disponibles en el laboratorio.

Ilustración 26: medida del voltaje regulado a través del potenciómetro.

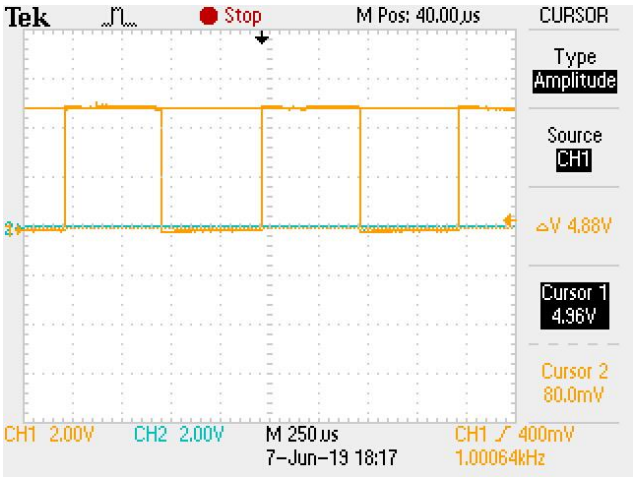

Ilustración 27: onda cuadrada generada usando el PLL de Philips Semiconductors.

En definitiva, los resultados de las medidas realizadas por la tarjeta de medida son:

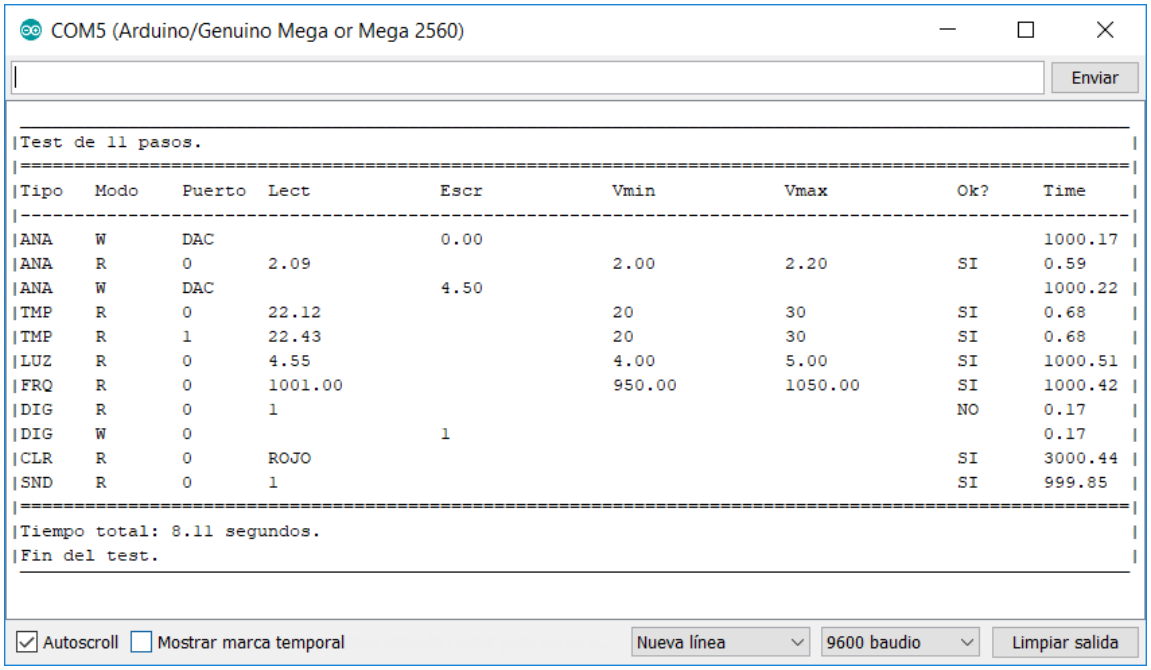

Ilustración 28: resultados obtenidos del test genérico.

Nótese que el resultado de la lectura digital es negativo, ya que la tarjeta de test (que maneja el sistema bajo pruebas) escribe un “1” lógico.

Y los resultados de las lecturas realizadas por la tarjeta de prueba, en los que se visualiza la lectura digital y la analógica cada vez que la tarjeta de medida (que maneja el sistema de test) realiza una escritura:

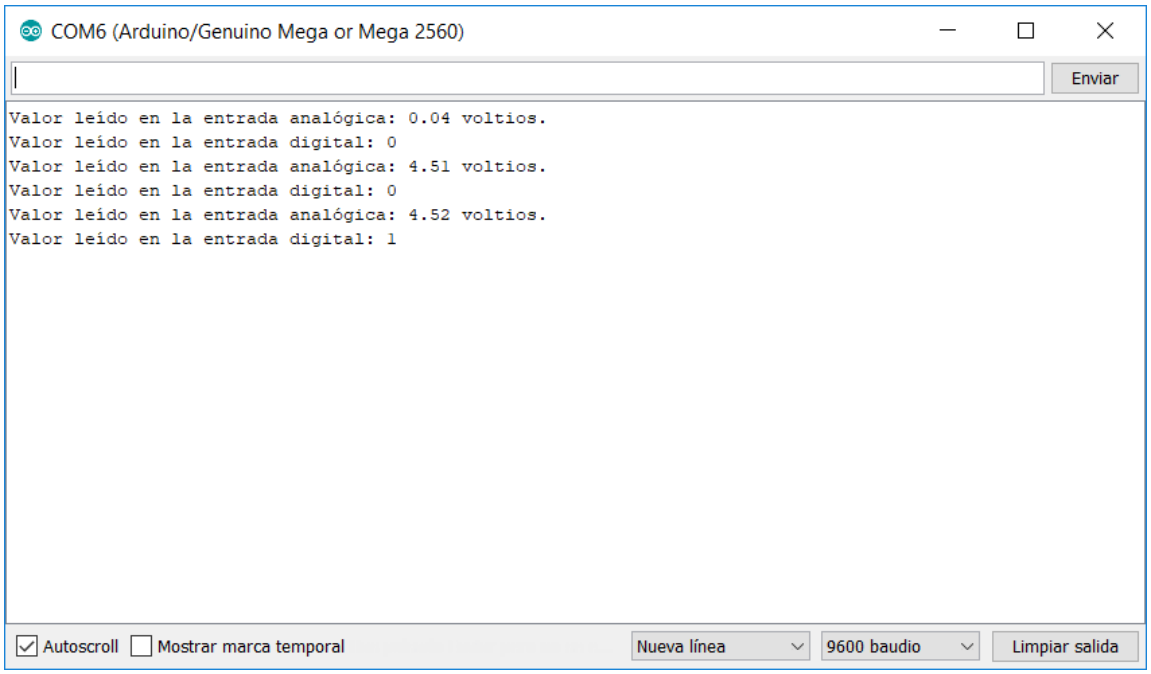

Ilustración 29: resultados mostrados a través de la tarjeta de test. Se muestra la lectura digital, así como la analógica realizadas, al principio del test, y cuando se realiza cada una de las escrituras.

8-Conclusiones del trabajo

Se dividirá el apartado final del trabajo en dos partes: por un lado, se comentarán los resultados del sistema construido. Por otro lado, se darán a conocer algunas posibles mejoras y extensiones del mismo.

8.1-Resultados del trabajo

·El resultado del presente Trabajo de Fin de Grado es plenamente satisfactorio: por una parte, el rendimiento del dispositivo de test implementado es muy bueno, y, además, se ha podido realizar dentro del tiempo establecido, pudiendo así alcanzar objetivos adicionales como son las mejoras del sistema de test.

Por otra parte, el desarrollo del proyecto pone en práctica conocimientos obtenidos durante todo el grado, tales como los fundamentos de programación, teoría de circuitos, electrónica analógica, gestión de proyectos, instrumentación electrónica y microelectrónica, entre otros. Ha sido especialmente útil para profundizar en la implementación de prototipos de sistemas electrónicos, sobre todo en el momento de montar el circuito, escribir la programación necesaria y realizar una posterior depuración, tanto en hardware como en software.

·Las medidas obtenidas utilizando ambas NTC han resultado coherentes entre sí, aunque no en periodos de tiempo extendidos. Un problema común en los termistores de coeficiente de temperatura negativo es el autocalentamiento debido a la disipación de potencia en el mismo. Este efecto puede ser especialmente perjudicial ya que, en algunos casos, las medidas obtenidas podrían ser erróneas a medida que pasa el tiempo desde que se encendió el sistema.

Sin embargo, este problema tiene una solución relativamente sencilla, que pasaría por alimentar estos componentes solo cuando sea necesario, haciendo que no pase corriente por los mismos. Es importante tener en cuenta, que, para una correcta implementación de esta solución, debe respetarse los límites de corriente a la salida en los pines digitales.

· Si bien es cierto que la respuesta entregada por dicho sensor es digital, el ajuste del umbral del mismo (es decir, el ajuste mediante el potenciómetro integrado en el sensor) es verdaderamente complicado, ya que para que se detecte el sonido generado por el zumbador, dicho umbral es tan preciso que, literalmente, es necesario aguantar la respiración al realizar el test. Por ello, y de cara a una implementación de mayor calidad del sistema, resultaría conveniente cambiar el modelo de micrófono utilizado.

Este presenta una respuesta bastante pobre con respecto a frecuencias audibles medias y altas, como puede ser, la del sonido generado por el *buzzer*, que viene controlado por el puerto digital **4** de la tarjeta de test. Concretamente, la frecuencia de la señal PWM generada es, en ese caso de 980 Hz (que puede ser medida a través del frecuencímetro implementado en el mismo sistema de test).

Cabe destacar, además, que el *buzzer* solo se utilizó para generar el sonido detectado por el micrófono en el test final mostrado en el apartado anterior. Durante el resto del desarrollo del proyecto, no se ajustó el micrófono tan precisamente. Así, el sonido emitido por el zumbador se

utilizó como señal, para notificar que debía generarse un sonido de otra forma, por ejemplo, un aplauso.

·El número de puertos, tanto digitales como analógicos ha hecho que el sistema no pudiera implementarse en muchos otros modelos. En total, en el circuito montado se han utilizado:

- ·2 pines de alimentación
- ·1 pin de referencia a tierra
- ·1 pin digital de escritura
- ·1 pin digital de lectura
- ·8 pines digitales como entradas del DAC
- ·1 pin digital para lecturas de frecuencia
- ·2 pines digitales para comunicación con la tarjeta de test
- ·5 pines analógicos para lecturas de temperatura, color, luminosidad y tensión.

Así, se estarían utilizando 14 pines digitales y 5 analógicos, que serían números muy ajustados para el modelo **Uno**, y que cuenta con 14 pines digitales y 6 pines analógicos [29]. Nótese que no se estaría contando con los pines necesarios para realizar la calibración del sensor de color.

Además, el montaje experimental del proyecto no utiliza todos los pines para los que la tarjeta de medida se programa: en total, está preparado para utilizar 14 pines analógicos para lecturas, y 23 pines digitales para diversas finalidades (entre ellas, la calibración del sensor de color).

Un detalle quizás más anecdótico, es la importancia de haber utilizado el modelo **Arduino MEGA**. De hecho, si se intenta compilar el programa para el modelo **Arduino UNO**, el IDE lanza un mensaje de error, ya que dicho modelo no cuenta con el registro **A** de pines digitales, que es utilizado por la función *calibraColor*. Si se cambia dicho registro por el registro **B** para que no dé errores, también se lanzará un mensaje de error, esta vez, notificando sobre la falta de espacio en la tarjeta **Arduino UNO** a la hora de cargar el programa.

·El cableado del circuito ha resultado ser bastante complejo, aunque el uso de la herramienta **Fritzing** ha sido de gran ayuda en ese aspecto, ya que permite diseñar gráficamente los circuitos sobre *breadboards*. Así, puede especificarse de antemano el código de colores, los elementos de circuito que se van a utilizar y su ubicación. Sin embargo, en el circuito que se ha implementado en el presente proyecto, ha sido necesario realizar algunas modificaciones con respecto al cableado, debido a la calidad media de las *breadboard* utilizadas, y que ha causado algún que otro problema, especialmente en la alimentación de los sensores.

·Aunque durante el desarrollo práctico del trabajo no se ha mostrado explícitamente, el modelo de **Arduino** utilizado presenta un gran rendimiento a la hora de medir frecuencias, llegando a medir frecuencias de señales a 1 MHz con un error razonable. La siguiente imagen muestra el un ensayo realizado en el laboratorio con un generador de señal, al que se conectó en serie un diodo para proteger el pin lector:

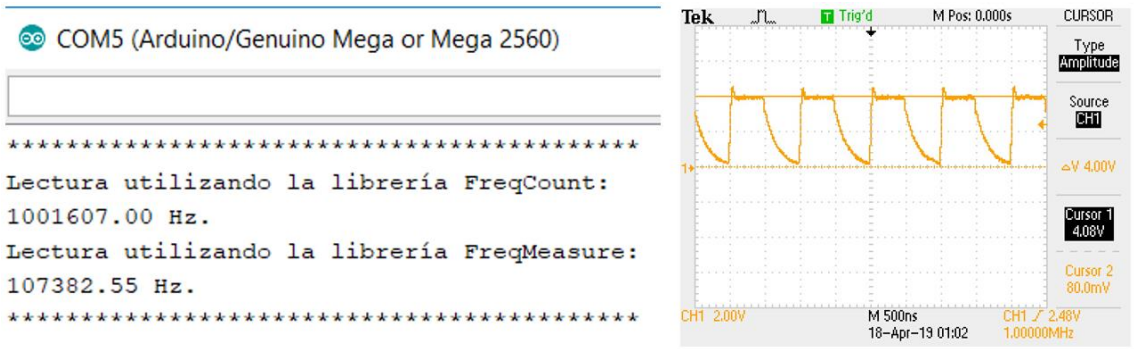

Ilustración 30: resultados obtenidos (izquierda) de la medida de frecuencia de una señal de 1 MHz (derecha). El efecto del diodo es notorio en el flanco descendente de la señal.

El error, usando la librería *freqCount* es de algo más de 1 kHz, lo cual podría ser inadmisible en algunos casos, pero relativamente, a 1 MHz es un error bastante bajo que se encuentra entre el 1 y el 2% de la frecuencia a medir. El resultado es satisfactorio teniendo en cuenta que se ha utilizado una tarjeta de desarrollo de bajo coste.

De igual manera, el resultado del conversor digital-analógico basado en la red de resistencias ha mostrado un buen rendimiento, especialmente si se tiene en cuenta la facilidad de su montaje. Su error, en principio, depende tan solo de la tolerancia de las resistencias utilizadas.

Con respecto a los sensores de sonido, luminosidad y color, el rendimiento es óptimo, ya que su funcionamiento está sujeto a calibraciones previas.

8.2-Posibles ampliaciones y mejoras del sistema

·Etapa de acondicionamiento para señales analógicas periódicas: puede construirse un sistema que adecúe las señales entrantes, para que las formas de onda en los puertos digitales de la tarjeta de medida sean ondas cuadradas con un ciclo de trabajo del 50%. Para ello, lo ideal sería empezar por filtrar el primer armónico de la señal. Después, la señal volvería a saturarse utilizando un amplificador operacional configurado como comparador, obteniendo así una señal cuadrada con el ciclo de trabajo adecuado (pero limitando el rango de frecuencias medibles).

El filtro tendría que diseñarse de modo que fuera sintonizable y de ganancia constante en un entorno, de modo que el armónico deseado siempre estuviera en su ancho de banda. Bajo estas consideraciones, el diseño de este sistema podría requerir conocimientos técnicos avanzados.

·Circuito impreso: una extensión interesante del proyecto podría ser la generación de una *PCB*, y que además supondría probablemente, una implementación más eficaz del sistema. Como pudo observarse anteriormente, las largas distancias en el cableado, y, sobre todo, los múltiples pasos por nodos de las *breadboard* pueden suponer una caída de potencial notoria. Cabe destacar que esta caída en el voltaje no se debe tanto a la longitud del conexionado, sino más bien a la calidad media de las *protoboard* utilizadas.

Además, una implementación sobre circuito impreso es realmente más robusta a pesar de su frágil apariencia, ya que los elementos quedan soldados sobre la placa. Por otra parte, el gran tamaño del sistema actual dejaría de ser un problema.

·Uso de la memoria no volátil: en el caso del modelo **Arduino MEGA 2560 R3**, la memoria EEPROM es de 4 kilobytes, tamaño lo suficientemente grande para guardar algunas variables de uso común y de especial relevancia: el número de bits del DAC, los coeficientes de temperatura de los termistores utilizados, y la lista de rangos de voltajes asociados a cada color en la salida del sensor de color, entre otras. Resultaría de especial utilidad en el sensor de color, ya que no habría que reescribir en el programa las líneas pertinentes con el fin de realizar las medidas correctamente tras la calibración.

·Debido a la complejidad del programa general utilizado en la tarjeta de medida, y al no haber usado interrupciones, las formas de onda generadas con el DAC (como, por ejemplo, la onda de diente de sierra generada para probar el funcionamiento del conversor) no son periódicas. Este hecho comprobarse con los osciloscopios disponibles en el laboratorio: la forma de onda mostrada en la pantalla se mueve en el eje horizontal al dibujar cada muestra incluso con el parámetro *trigger* correctamente ajustado. Una opción interesante podría ser el uso de dichas interrupciones contenidas en el lenguaje de programación utilizado, y que podría llevar a generar señales de varias formas.

·Caracterización de sistemas en frecuencia: podría incluirse, como complemento al sistema de test implementado, otro sistema destinado a la caracterización en frecuencia. Para ello, sería necesario un generador de ondas sinusoidales. Podría utilizarse un filtro paso banda sintonizable, junto con un generador de ondas cuadradas (algún VCO común). Además de la implementación del mencionado hardware, también sería necesario, muy probablemente, un uso extensivo de las interrupciones de **Arduino**. Para generar señales sin *offset,* se necesitaría un inversor de tensión, que podría exigir, según el caso, demasiada corriente a las tarjetas de desarrollo.

Referencias

[1] U. Z. D. Gaiero, «ICT vs FCT Test: case studies,» SPEA, 2014.

[2] J. Bateson, «9.5-Test System Hardware,» de *In-Circuit Testing*, New York, Springer, 1985, pp. 192-195.

[3] Feinmetall, «Different Types of Spring Contact Probes,» de *Contact Probes for PCB Testing*, 2017, p. 7.

[4] I. Poole, «ICT, In Circuit Test Fixtures / Bed-of-Nails & Probes,» [En línea]. Available: https://www.electronics-notes.com/articles/test-methods/automatic-automated-test-ate/ict-in-circuit-test-fixture-bed-of-nails-probes.php. [Último acceso: 23 2 2019].

[5] T. A. L. D. Grahame Holmes, «Fundamental Concepts of PWM,» de *Pulse Width Modulation For Power Converters: Principles and Practice*, Wiley-Interscience, IEEE Press, 2003, pp. 95-96.

[6] S. M. John Park, «Digital to analog converters,» de *Practical Data Acquisition for Instrumentation and Control Systems*, Elsevier, 2003, p. 159.

[7] M. Anand, «Digital Meters,» de *Electronic Instruments and Instrumentation Technology*, PHI Learning Pvt., 2004, pp. 58-60.

[8] R. P. Areny, «Fotorresistencias (LDR),» de *Sensores y acondicionadores de señal*, Marcombo, 2006, pp. 88-89.

[9] RS Components, «RS Components | Componentes Electrónicos y Eléctricos,» [En línea]. Available: https://es.rs-online.com/web/p/resistores-dependientes-de-luz-ldr/9146710/. [Último acceso: 2 6 2019].

[10] J. G. Webster, «Acoustic Measurements,» de *The Measurement, Instrumentation and Sensors Handbook*, IEEE Press, CRC Press, 1999.

[11] RS Components, «RS Components | Componentes Electrónicos y Eléctricos,» [En línea]. Available: https://es.rs-online.com/web/p/termistores/1912263/.

[12] J. G. Webster, «Thermistor Thermometers,» de *The Measurement, Instrumentation and Sensors Handbook*, CRC Press, IEEE Press, 1999.

[13] ELEGOO, «Lesson 23: Thermometer,» de *The most complete starter kit for MEGA 2560*, p. 160.

[14] R. P. Areny, «Termistores,» de *Sensores y acondicionadores de señal*, Marcombo, 2006, pp. 72-74.

[15] J. G. Webster, «Photojunction sensors,» de *The Measurement, Instrumentation and Sensors Handbook*, CRC Press, IEEE Press, 1999.

[16] RS Components, «RS Components | Componentes Electrónicos y Eléctricos,» [En línea]. Available: https://es.rs-online.com/web/p/fototransistores/6548542/. [Último acceso: 8 6 2019].

[17] N. G. K.M. Gupta, «9-Optoelectronic Devices,» de *Advanced Semiconducting Materials and Devices*, Springer International Publishing, 2016, pp. 341,342.

[18] OSRAM Opto Semiconductors, «Silicon NPN Phototransistor, Version 1.3, SFH 310,» 2016.

[19] Arduino Store, «Arduino Mega 2560 Rev3,» [En línea]. Available: https://store.arduino.cc/mega-2560-r3. [Último acceso: 2 4 2019].

[20] A. Piganti, «DIWO | Do It With Others. La escuela Maker de bq,» 2014 11 25. [En línea]. Available: http://diwo.bq.com/pinout-mega/. [Último acceso: 1 6 2019].

[21] PJRC: Electronic Projects, «FreqCount Library,» [En línea]. Available: https://www.pjrc.com/teensy/td_libs_FreqCount.html. [Último acceso: 5 4 2019].

[22] PJRC: Electronic Projects, «FreqMeasure Library,» [En línea]. Available: https://www.pjrc.com/teensy/td_libs_FreqMeasure.html. [Último acceso: 5 4 2019].

[23] Microchip Technology Inc., «MCP601/1R/2/3/4, 2.7V to 6.0V Single Supply CMOS Op Amps,» 2007.

[24] EPCOS, «NTC Thermistors for temperature measurement. Leaded NTC thermistors, lead spacing 5 mm. Series/type: B57164K,» 2013.

[25] LUNA Optoelectronics, «To-18 Ceramic package photocells, NSL-19M51,» Camarillo, 2016.

[26] O. Torrente, «Emisión de sonido, uso de zumbadores,» de *Arduino. Curso práctico de formación*, Madrid, RC Libros, 2013, p. 365.

[27] Texas Instruments, «xxx555 Precision Timers,» 2014.

[28] Hardware Hacks, «Bubble Sorting with a Arduino/C++ Application,» 3 5 2016. [En línea]. Available: http://hwhacks.com/2016/05/03/bubble-sorting-with-an-arduinoc-application/. [Último acceso: 2019 6 12].

[29] A. Piganti, «DIWO | Do It With Others. La escuela Maker de bq,» 2014 11 25. [En línea]. Available: http://diwo.bq.com/pinout-uno/. [Último acceso: 8 6 2019].

[30] H. Camenzind, «11-Timers and Oscillators,» de *Designing Analog Chips*, Virtualbookworm.com Publishing, 2005, p. 2.