UNIVERSIDAD AUTÓNOMA DEL
ESTADO DE QUINTANA ROO

DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

ESTUDIO MONOGRÁFICO DE SENSORES PARA SISTEMAS ARDUINO Y EMBEBIDOS.

TESIS

PARA OBTENER EL GRADO DE

INGENIERO EN SISTEMAS DE ENERGÍA

PRESENTA

JULIA CECILIA UH US

ASESORES
Dr. FREDDY IGNACIO CHAN PUC
Dr. EMMANUEL TORRES MONTALVO
Dr. HOMERO TORAL CRUZ
MM. JESUS ORIFIEL ALVAREZ RUIZ
MM. CRISTIAN ANZURES MENDOZA

CHETUMAL QUINTANA ROO, MÉXICO, OCTUBRE DE 2017

UNIVERSIDAD AUTÓNOMA DEL
ESTADO DE QUINTANA ROO

DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

TRABAJO MONOGRÁFICO TITULADO

"ESTUDIO MONOGRÁFICO DE SENSORES PARA SISTEMAS ARDUINO Y EMBEBIDOS."

ELABORADO POR

Julia Cecilia Uh Us

BAJO SUPERVISIÓN DEL COMITÉ DEL PROGRAMA DE LICENCIATURA Y APROBADO COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE:

INGENIERO EN SISTEMAS DE ENERGÍA

COMITÉ SUPERVISOR

SUPERVISOR: Dr. FREDDY IGNACIO CHAN PUC

SUPERVISOR: Dr. EMMANUEL TORRES MONTALVO

SUPERVISOR: Dr. HOMERO TORAL CRUZ

SUPERVISOR SUPLENTE: MM. JESUS ORIFIEL ALVAREZ RUIZ

SUPERVISOR SUPLENTE: MM. CRISTIAN ANZURES MENDOZA.

ÁREA DE TITULACIÓN

CHETUMAL QUINTANA ROO, MÉXICO, DE OCTUBRE DE 202

DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

Resumen

Presentamos este estudio monográfico como su nombre lo indica un estudio de sensores básicos y de última generación para sistemas Arduino y embebidos. Los sensores que citamos son desde los básicos, los más usados, económicos y accesibles, ya que están dirigidos desde uso de proyectos más simples hasta los más complejos.

Se utiliza el sistema Arduino uno y los sensores compatibles ya que son fáciles de usar, ya sea con la ayuda de tu profesión o si eres profesor de nuevas tecnologías lo puede incluir en las clases tanto profesional como particular. También se debe a su popularidad y gran diversidad.

Las razones por las que Arduino es bueno, se debe a que nos permite desarrollar proyectos de bajo costo y flexibles, es decir, nos permite trabajar en casi todas las plataformas informáticas, por no decir todas, desde Mac OS X a Linux, pasando por Windows. También hay flexibilidad debido a que dependiendo del proyecto que tengas en mente hay un Arduino para el proyecto.

En cuanto al lenguaje de programación empleamos El Entorno de Desarrollo Integrado IDE en el cual podemos escribir el código de forma fácil, con muchas ayudas y herramientas. Es un software que está en constante actualización.

Se analizaron los esquemas de conexión, librerías y ejemplos de códigos de programación para el funcionamiento de los diferentes tipos de sensores desde el sensor ultrasónico que es básico hasta el sensor detector de polvo, sensor detector de humedad y temperatura que son los avanzados y que son compatibles con la palca Arduino.

Se espera que este trabajo sea una guía para el uso de los sensores en otros trabajos de aplicación.

Agradecimientos

Este es otro año más en el que debo reconocer la bondad de Dios en mi vida. Has sido mi consuelo, mi esperanza y mi fortaleza. Te doy gracias, Dios, por estar siempre conmigo en mi vida.

Agradezco a mis profesores Dr. Bojórquez Báez Inocente, Dr. José Hernández, Dr. Yam Gamboa Joel Omar, M.E.S Roberto Acosta, a mi tutor Emmanuel Torres Montalvo, + Dr. Fernando flores Murrieta que formaron parte en mi desarrollo profecional académico, que sin su apoyo y conocimiento no estaría donde me encuentro ahora.

Un agradecimiento especial a mi director de tesis y profesor al Dr. Freddy Ignacio por brindarme la oportunidad Chan Puc de quien tuve el disponibilidad, apoyo en todo momento en este proyecto de investigación.
Gracia a mis amigos y compañeros de generación por compartir conocimientos y experiencias.

A mis padres Pascual B. Uh Yah y Cecilia Us Camargo que con amor y cariño me educaron con sentido humano, me cultivaron el gusto por aprender y salir adelante. A mi esposo Ismael Arturo por su amor y su apoyo incondicional.
Este logro más que mío es de todos aquellos que lo hicieron posible y jamás me alcanzarán las palabras para agradecerles por tanto.

Dedicatoria

Primeramente les dedico este sueño hecho realidad a mis padres Pascual B. Uh Yah y +Cecilia Us Camargo que con su ejemplo desde muy pequeña me enseñaron el significado de perseverar y luchar por nuestros sueños, a mis hermanos Lizardo Uh Us, Freddy UH Us, Marianiza Uh Us, Roció Uh Us, sarita Uh Us porque fueron mis primeros compañeros de vida y me enseñaron lo esencial que es un equipo.

A mi amado esposo Lic. Ismael Arturo López Bardales que me impulso para seguir adelante en cada caída o dificultad, le agradezco porque siempre creyó en mí y me hizo confiar en mí misma.

A mis dos hermosos hijo Santiago Ismael y Julián Arturo que son mi motor e inspiración en la vida y la razón para seguir adelante.

A mi director de tesis Dr. Freddy Chan Puc. Por bridarme la oportunidad de desarrollar este trabajo de investigación.

¡A Dios por su don inefable!

Contenido de temas

Contenido de Figuras

Contenido de Tablas

ANTECEDENTES.

Arduino fue inventado en el año 2005 por el entonces estudiante del instituto IVRAE Massimo Banzi, quien, en un principio, pensaba en hacer Arduino por una necesidad de aprendizaje para los estudiantes de computación y electrónica del mismo instituto, ya que en ese entonces, adquirir una placa de microcontroladores eran bastante caro y no ofrecían el soporte adecuado; no obstante, nunca se imaginó que esta herramienta se llegaría a convertir en años más adelante en el líder mundial de tecnologías DIY (Do It Yourself). Inicialmente fue un proyecto creado no solo para economizar la creación de proyectos escolares dentro del instituto, sino que además, Banzi tenía la intención de ayudar a su escuela a evitar la quiebra de la misma con las ganancias que produciría vendiendo sus placas dentro del campus a un precio accesible.

El primer prototipo de Arduino fue fabricado en el instituto IVRAE. Inicialmente estaba basado en una simple placa de circuitos eléctricos, donde estaban conectados un microcontrolador simple junto con resistencias de voltaje, además de que únicamente podían conectarse sensores simples como leds u otras resistencias, y es más, aún no contaba con el soporte de algún lenguaje de programación para manipularla. (Arduino, s.f.)

Tiempo después, se integró al "Team Arduino" el estudiante español David Cuartielles, experto en circuitos y computadoras, quien ayudó Banzi a mejorar la interfaz de hardware de esta placa, agregando los microcontroladores necesarios para brindar soporte y memoria al lenguaje de programación para manipular esta plataforma.

Un breve tiempo más tarde, al ver los grandes resultados que tuvo Arduino y las grandes aceptaciones que tuvo por parte del público, comenzó a distribuirse en Italia, después en España, hasta colocarse en el número uno de herramientas de aprendizaje para el desarrollo de sistemas autómatas, siendo además muy económica ($300-500) en comparación con otras placas de microcontroladores ($800 en adelante).

Google ha apostado por el proyecto y ha colaborado en el Android ADK (Accesory Development Kit), una placa Arduino capaz de comunicarse directamente con Smartphone, Android para obtener las funcionalidades del teléfono (GPS, acelerómetros, GSM, a bases de datos) y viceversa para que el teléfono controle luces, motores y sensores conectados de Arduino.

El estudiante colombiano Hernando Barragán, fue quien desarrolló la tarjeta electrónica Wiring, el lenguaje de programación y la plataforma de desarrollo. Basándose en su trabajo, Massimo, David Cuartilles, investigador en el instituto, y Gianluca Martino desarrollador, desarrollaron una plataforma de hardware y software libre, más pequeño y económico, a la que llamaron Arduino.

El nombre del proyecto, tiene su origen en el Bar di Re Arduino (Antiguo Rey de Italia entorno al 1002), donde Massimo Banzi, uno de los fundadores, pasaba parte de su tiempo libre.

Poco tiempo después de terminar el desarrollo, el instituto cerró sus puertas y los desarrolladores intentaron sobrevivir con el nuevo sistema Arduino.

El proyecto gustó mucho, desplazando a otras soluciones del mercado como Basic Stamp y los míticos Pics. El mismo Google colaboró en el desarrollo del Kit ADK (Accesory Development Kit), una placa Arduino capaz de comunicarse directamente con teléfonos móviles inteligentes bajo el sistema operativo Android.

Para la producción en serie de la primera versión, se buscó no superar el precio de $360.00 y que se ensamblara en una placa azul y que fuera como plug and play y compatible con múltiples sistemas operativos: Mac OSX, Windows y GNU/Linux. Las primeras 300 unidades se las dieron a los alumnos del Instituto IVREA, con el fin de que las probaran y empezaran a diseñar sus primeros prototipos. El IDE de Arduino se desarrolló basándose en Processing, buscando la sencillez y la portabilidad a múltiples sistemas operativos. (Sixto, s.f.)

En la feria Maker Fair de 2011 se presentó la primera placa Arduino de 32 bit para trabajar tareas más pesadas, y llegaron nuevas contribuciones al proyecto de parte de Intel, con su placa Galileo.

Tras el enorme éxito del proyecto, aparecieron clones, compatibles y sistemas similares, basados en otros microcontroladores, como Pingüino, basado el PIC 18F. Incluso el propio fabricante de los PIC, microchip, lanzó chipKIT, con PIC32 compatible con el hardware y el software de Arduino.

Un breve tiempo más tarde, al ver los grandes resultados que tuvo Arduino y las grandes aceptaciones que tuvo por parte del público, comenzó a distribuirse en Italia, después en España, hasta colocarse en el número uno de herramientas de aprendizaje para el desarrollo de sistemas autómatas, siendo además muy económica entre $300.00 - $500.00 en comparación con otras placas de micro controladores ($800.00 en adelante). (INDUSTRUINO)

CAPÍTULO 1

INTRODUCCIÓN AL SISTEMA ARDUINO.

La plataforma Arduino es cada vez más popular y ofrece gran diversidad de placas sobre las que se pueden programar nuestros proyectos, ya que existen multitud de ellas.

Basada en la filosofía del software libre, Arduino es una plataforma de electrónica «open-source» o de código abierto cuyos principios son contar con software y hardware fáciles de usar. Básicamente lo que permite esta herramienta es la generación de infinidad de tipos de microordenadores de una sola placa, que luego pueden tener una amplia variedad de usos según la necesidad de la persona que lo utiliza. Es decir, una forma sencilla de realizar proyectos interactivos para cualquier persona. (Fundación)

Entonces, ¿te imaginas ya para qué sirve un Arduino? Por darte una idea, con un Arduino puedes crear básicamente lo que quieras, desde una báscula, un reloj, hasta unas puertas controladas por voz, etc. Para que puedas entender cómo podemos pasar de un microordenador a un sistema complejo como el que acabamos de mencionar, vamos a profundizar en qué son las placas Arduino.

Arduino se puede utilizar para desarrollar elementos autónomos, o bien conectarse a otros dispositivos o interactuar con otros programas, para interactuar tanto con el hardware como con el software. Sirve tanto para controlar un elemento, pongamos por ejemplo un motor que nos suba o baje una persiana basada en la luz que haya gracias a un sensor conectado al Arduino, o bien para transformar la información de una fuente, como puede ser un teclado, y convertir la información a algo que entienda, por ejemplo, un ordenador.

Actualmente, el uso de Arduino puede catalogarse en dos grandes grupos:

1. Arduino es utilizado como un microcontrolador, cuando tiene un programa descargado desde un ordenador y funciona de forma independiente de éste, y controla y alimenta determinados dispositivos y toma decisiones de acuerdo al programa descargado e interactúa con el mundo físico gracias a sensores y actuadores.
2. Arduino hace de interfaz entre un ordenador u otro dispositivo, que ejecuta una determinada tarea, para traducir dicha tarea en el mundo físico a una acción.

Y viceversa, gracias a sensores que están conectados a la placa Arduino podemos hacer que el ordenador ejecute determinada acción.

Arduino es una de las placas más populares del mundo. Su versatilidad y la infinidad de posibilidades que ofrece la convierten en una de las herramientas de programación más completas del mercado. (Elecronic, 2021)

El proceso del prototipo con Arduino

Arduino es una placa orientada al prototipo. No sólo porque todo lo que está en la placa y el software están pensados para prototipar (versión inicial de la idea de un producto o servicio) de una forma muy fácil y rápida. Además, tenemos sensores, actuadores y shields alrededor de Arduino que nos facilitan esta tarea.

Cuando tenemos que empezar a crear un proyecto con Arduino lo primero que tenemos que hacer es plantear ese proyecto. Debemos plasmarlo con lápiz o bolígrafo en un papel. Así de sencillo.

Por eso es recomendable que sigas estas 3 fases.

1. La idea general del proyecto con Arduino:

Iniciar con la imaginación y plasmar la idea general con todos los componentes que se te ocurran.

Si lo que estás planteando, por ejemplo, es hacer un proyecto con Arduino para medir la temperatura, en esta fase el proyecto no tiene que ser simple, no te cortes y plasma todo en un simple papel, realiza un dibujo sencillo, sin complicaciones, donde vengan todos los componentes que te gustaría tener en tu proyecto. Al final de esta fase tendrás un listado de ideas y especificaciones del proyecto, Este sería el inicio de la siguiente fase, el prototipo mínimo.

2. El prototipo mínimo

Ya se tienes ese maravilloso proyecto con Arduino en un papel. Junto a ese diseño tienes un listado de ideas y especificaciones. Ahora tienes que pensar de tu yo más racional, digamos debes plantearte una pregunta crucial, debe centrarse en el objetivo principal de tu proyecto**.**

Esta fase consiste en eliminar todo aquello que no es importante y que solo aportan funcionalidades extras.

En concreto, consigue crear un prototipo mínimo que sea capaz de hacer la funcionalidad principal de tu proyecto.

Debes ser capaz de conocer tus limitaciones y de los medios que dispones. Es simple, si consigues hacer un prototipo mínimo en poco tiempo, no sentirás la frustración de no llevar tu idea a la realidad. Esta fase es crucial para aprender Arduino.

Debes ser capaz de con muy poco, obtener resultados**.** Siempre tendrás tiempo de ir mejorando el proyecto con Arduino y añadir nuevas funcionalidades.

3. El diseño incremental

Ahora sí, ya tienes las ideas claras de lo que tienes que hacer para llevar a cabo tu prototipo mínimo. Lo has plasmado todo y ahora empieza la fase de creación. Ya puedes empezar a crear los circuitos y a programar Arduino a partir de ese prototipo mínimo.

Mi consejo es que apliques el diseño incremental. Básicamente consiste en dividir ese planteamiento que vas a resolver, en pequeños problemas. Haz un bosquejo de lo que quieres conseguir y poco a poco vas dando forma a esa idea.

Plantea y reflexiona lo que tienes que hacer en el código, investiga sobre las características del hardware y el software. Solo cuando tengas toda esta información deberás ejecutar las acciones necesarias para realizar tu prototipo mínimo. (Mastoner, 2022)

Arduino: Introducción al software y hardware

Dos disciplinas que debes dominar en un proyecto de Arduino son la programación y la electrónica. No queda de otra, que no puedes pasar por esto.

Pero al contrario que otras tecnologías como pueda ser un ratón o un teclado, denominadas tecnologías *Plug&Play*, Arduino no es un hardware que se conecte y listo. Con los proyectos para Arduino debes aprender todas las nociones para que puedas configurar y programar el microcontrolador de Arduino.

Para empezar, debes conocer el software y el hardware que está involucrado. (Fernández, 2022)

A. Entorno de desarrollo de Arduino

Para realizar proyectos con Arduino tiene que programar es la única manera de llevar a cabo sus proyectos, Esta programación, ya sea para Arduino, para otro tipo de placa o para otro lenguaje de programación, se suele hacer a través de un IDE o entorno de desarrollo. Pero, ¿qué es un IDE o entorno de desarrollo?

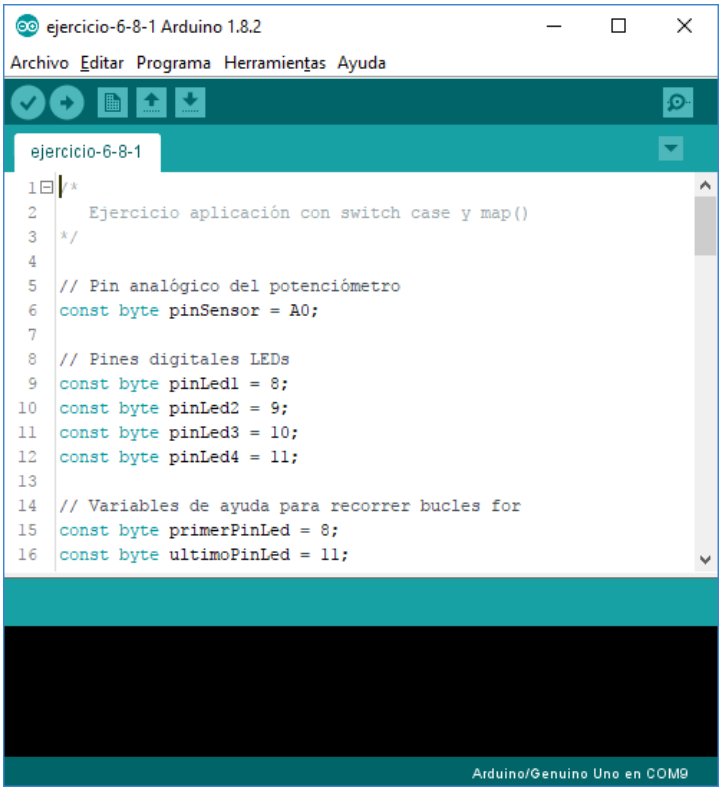
Le explicamos con una analogía. Cuando está escribiendo un informe o cualquier documento para tu trabajo ¿con qué software lo haces? Normalmente se utiliza Office de Microsoft o la versión de código abierto Libre Office.

Pero también lo podrías hacer con un Bloc de Notas. ¿Qué ventajas tiene escribir un documento de este estilo con un programa como Word?

Un procesador de texto potente te permitirá añadir tablas, utilizar listas y demás herramientas que facilitan el crear un documento. Seguramente todo esto no lo puedas hacer con el Bloc de Notas de Windows por ejemplo.

Con los entornos de desarrollo ocurre algo parecido. Con el IDE de Arduino podemos escribir nuestro código de una forma muy fácil y sobre todo, con muchas ayudas o herramientas.

Figura 1. IDE de Arduino escritura de un código.



Fuente: Sistema Arduino

Por ejemplo, cuando escribimos una palabra reservada nos la cambia de color. Podemos contraer estructuras de control o funciones. Insertar y gestionar librerías a través del menú o formatear el código.

Pero lo más importante del IDE de Arduino es que podemos cargar el código a la placa. Podríamos escribir todo nuestro código en el Bloc de Notas o algún otro editor de texto y sería totalmente válido. El código no es más que texto.

Sin embargo, el IDE de Arduino nos permite hacer una cosa fundamental: compilar el código y subirlo a la placa. Esa es la clave.

Otra característica importante del IDE de Arduino es que es de código abierto. Es decir, Arduino es una plataforma abierta. Dentro de esta plataforma se incluye el IDE de Arduino.

Esto significa que ya no solo tenemos el software de forma gratuita, también lo podemos modificar a nuestro antojo. Eso sí, para hacer cualquier mejora o cambio debes conocer el lenguaje de programación con el que se programa el propio entorno de desarrollo de Arduino, IDE.

No confundir tecnologías libres con tecnologías gratuitas. Aunque una cosa lleve a la otra, no quita que detrás de este tipo de proyectos haya mucha gente trabajando y muchas horas de dedicación. Hay diferentes formas. Si nos centramos en Arduino la más sencilla es comprar placas originales y no copias. Pero también puedes ayudar haciendo una donación. Eso servirá para pagar el hosting de la página web, a los trabajadores o cualquier gasto derivado del proyecto.

Versiones del IDE de Arduino

Se trata de un software que está en constante actualización. Arduino no tiene un periodo fijo a la hora de hacer actualizaciones. Cuando sale una nueva versión se añaden nuevas opciones o se corrigen errores.

Lo más importante es que de momento (vamos por la versión 1.8.6) el IDE sigue manteniendo su mismo aspecto. Esto es más importante de lo que parece. Las opciones suelen estar siempre en el mismo sitio y por lo tanto, cuando se actualiza apenas notarás la diferencia.

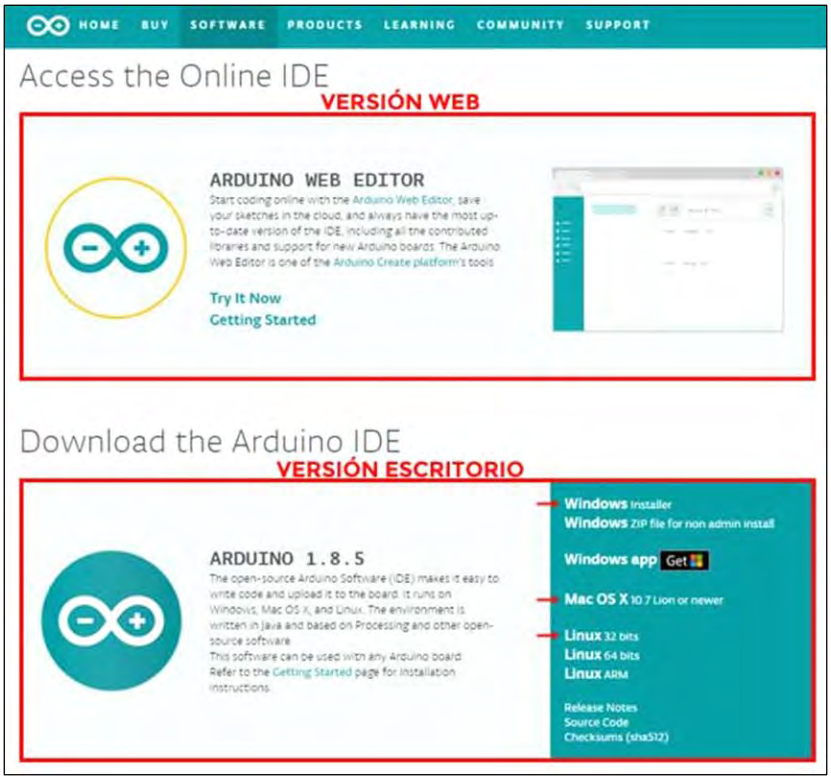
Pero por ahora tampoco cambia el código. Esto quiere decir que si tienes un programa que hiciste con la versión 1.4 de Arduino, también te servirá para cargarlo a la placa con la versión 1.8. (Hernández)

B. Instalación del IDE de Arduino

El segundo paso, una vez visto qué es un IDE y qué nos puede aportar, es la instalación. Podemos decir que este es el primer paso técnico para empezar a realizar el proyecto de Arduino. Todo parte de aquí.

Accede a la sección de software de Arduino. En la actualidad podemos programar de dos formas a través del IDE oficial de Arduino. Hay una versión web y una versión de escritorio. Para empezar te recomiendo que lo hagas a través de la versión de escritorio.

Figura 2. IDE oficial de Arduino versión web y versión de escritorio.



Fuente: Sistema Arduino

Debe elegir la versión para su sistema operativo. Hay una versión para Windows, para Linux o para Mac OS X.

Todo lo demás que continúa a partir de aquí es una secuencia de pantallas que lo único que hacen es instalar todo lo necesario para poder programar con el IDE de Arduino. A continuación, un resumen rápido de la secuencia de instalación.

En este caso se realiza para Windows, ya que hasta el día hoy es el sistema operativo más utilizado del mundo. (Hernández)

C. Funciones principales del IDE de Arduino

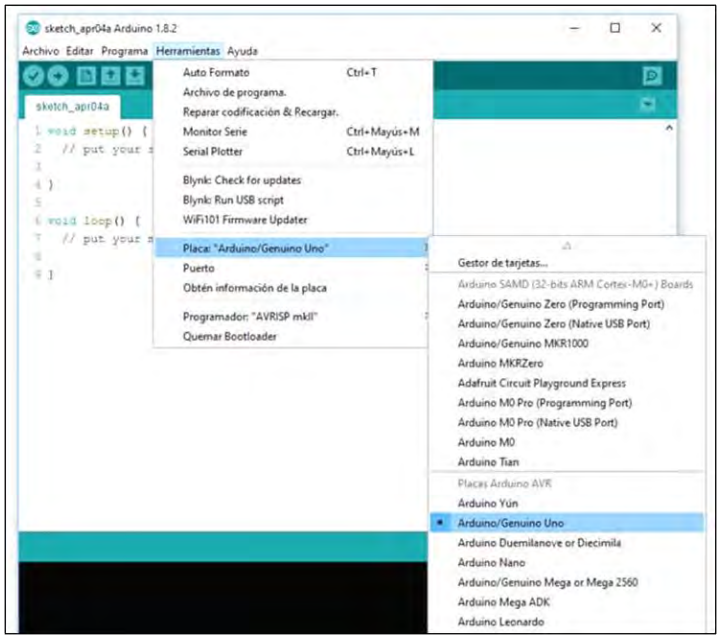
En esta sección le enseñamos las funciones más importantes del IDE de Arduino. Aprenderás lo básico para empezar a programar tus propios programas. Aprender Arduino requiere de este fuerzo previo por conocer las herramientas. No conseguirás avanzar rápido si antes no dominas los conceptos básicos del software y hardware de Arduino.

Seleccionar la placa correcta y el puerto serie:

Seleccionar la placa es relativamente sencillo. Con el IDE podemos trabajar con todos los modelos de Arduino e incluso con modelos que no son de la misma marca. Un ejemplo es el ESP8266.

Puedes seleccionar la placa a través del menú en herramientas>placas >Arduino Uno. No hace falta que conectes al ordenador para poder seleccionar el modelo de la placa a utilizar.

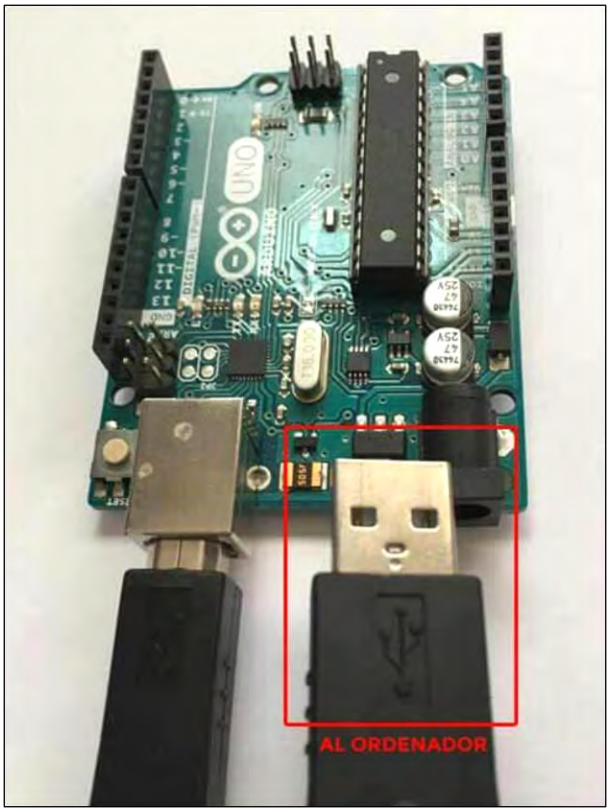
Figura 3. IDE de Arduino seleccionando el tipo de placa.



Fuente: Sistema Arduino

El puerto serie es por donde se comunican Arduino y el ordenador. Es necesario que tengas conectado tu Arduino al ordenador. Es muy sencillo, no tiene pérdida.

Figura 4. placa Arduino Uno muestra el puerto serie USB que conecta al ordenador.



Fuente: Sistema Arduino

Para seleccionar el puerto lo hacemos a través del menú *Herramientas>Puerto*. Puede que aparezca más de uno y además el nombre varía según el sistema operativo. (Hernández)

Preferencias del sistema

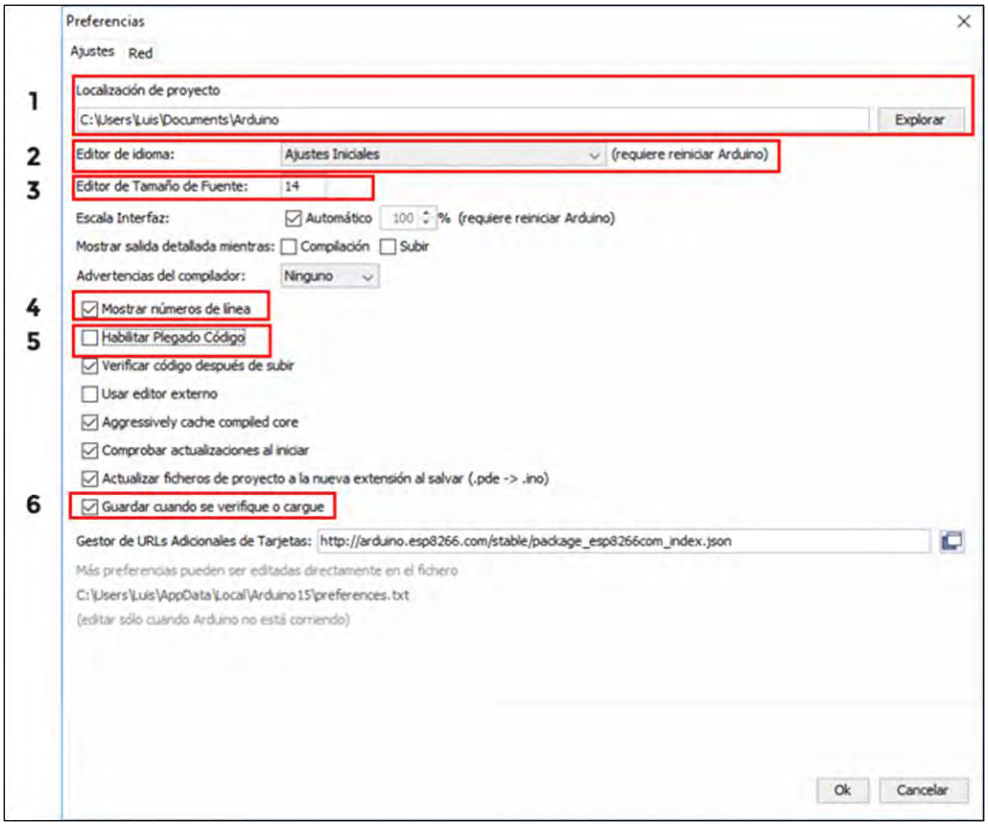
Como en casi todos los programas que utilizamos, en el IDE de Arduino tenemos una opción para configurar las preferencias del sistema. Nos permite modificar el idioma, el tamaño de letra y algunas cosas más que veremos.

Para acceder a esta opción solo tenemos que ir al menú archivo>*Preferencias*.

Presentamos las principales funciones que nos permiten modificar el aspecto y funcionamiento del IDE de Arduino:

- Localización del proyecto: podemos seleccionar una carpeta donde iremos guardando los proyectos. Por defecto será la que ha creado el instalador en documentos/Arduino. Esta ruta varía según el sistema operativo.
- Editor de idioma: con esta opción podemos cambiar el idioma del IDE.
- Editor de Tamaño de Fuente: indica el tamaño de fuente del editor del IDE.
- Mostrar número de línea: para que muestre los números de líneas en el editor.
- Habilitar plegado el código: siempre que el código tenga una sentencia con {} nos permitirá contraer y expandir ese código. Muy útil cuando trabajamos con archivos muy grandes.
- Guardar cuando se verifique o cargue: es importante que cuando verifiquemos el código o lo carguemos al microcontrolador haga un guardado automático. Déjalo marcado. (Crespo, 2017)

Figura 5. Selección de las opciones y funcionamientos más importantes el IDE



Fuente: Sistema Arduino

Sistema de ficheros de Arduino

Una de las mejoras que han ido introduciendo dentro del IDE de Arduino es la gestión de archivos. Lo primero que debes conocer es la extensión con que se guardan los ficheros de Arduino, .ino.

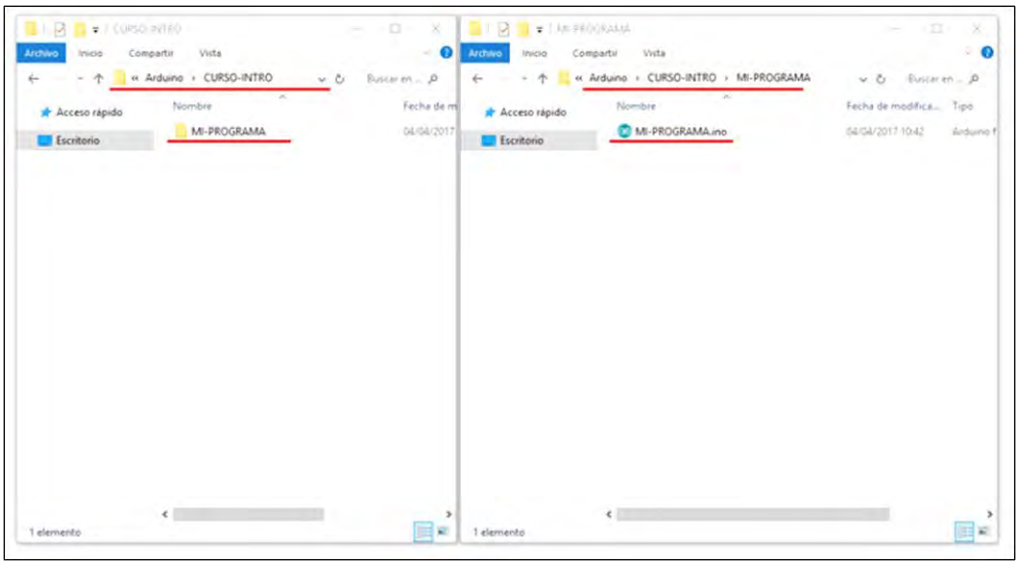
Si has creado un programa o sketch (sketch significa esquema o bosquejo) verás que tiene una extensión .ino.

Cuando guardas un archivo en tu ordenador, el propio IDE de Arduino ya lo organiza por ti. Crea una carpeta con el mismo nombre que el archivo y dentro guarda el fichero.

Por ejemplo, si creas un nuevo programa y vas al menú *Archivo>Salvar*, te permitirá guardarlo con un nombre.

Y luego, en la ruta que hayas elegido habrá creado una carpeta con el mismo nombre y dentro el fichero. (Crespo, 2017)

Figura 6. Ejemplo de realizar una carpeta con el mismo nombre y dentro el fichero

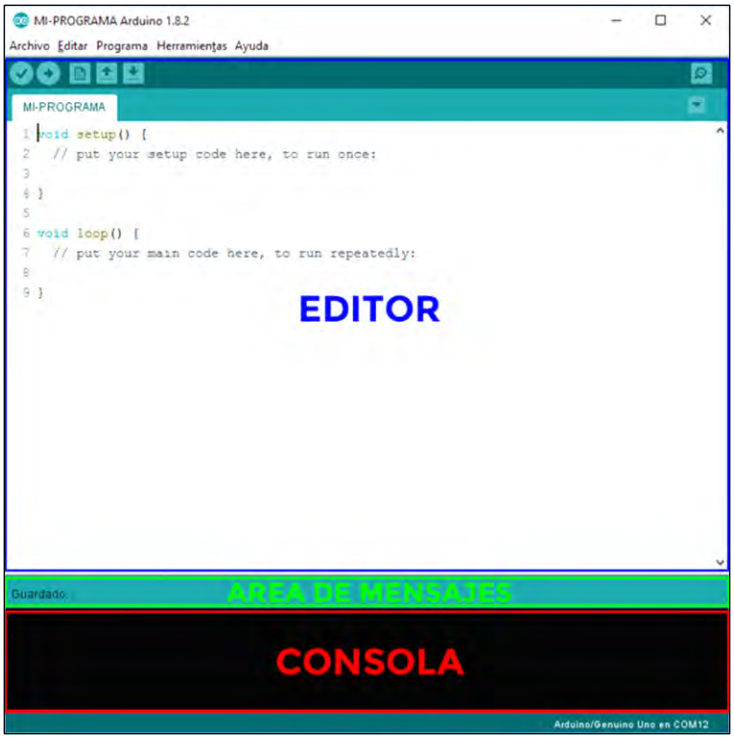


Fuente: Sistema Arduino

Partes fundamentales del IDE de Arduino.

Dentro del IDE de Arduino podemos destacar 3 partes principales: El editor, el área de mensajes y la consola. El editor:

Figura 7. Editor parte fundamentales del IDE de Arduino.

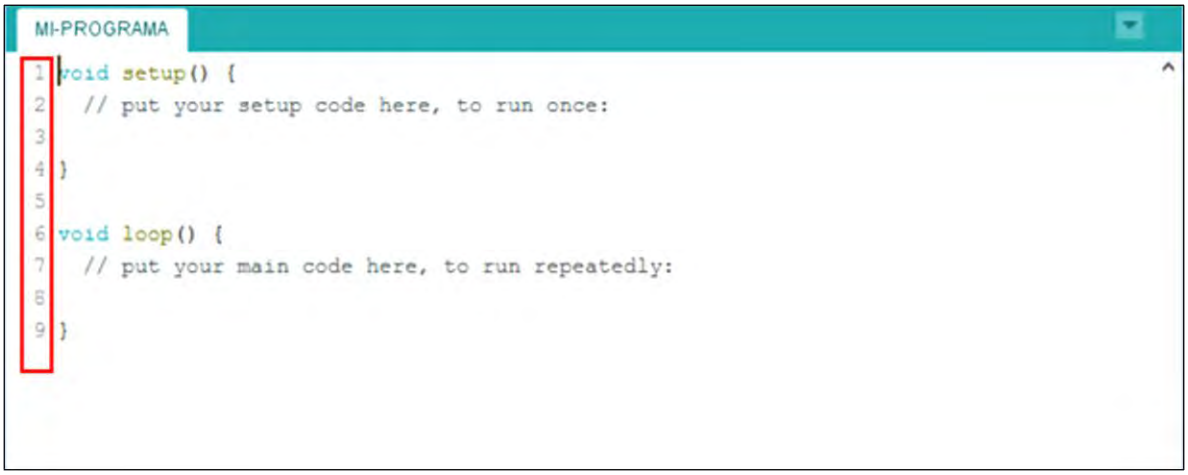


Fuente: Sistema Arduino

Aquí es donde más vamos a trabajar ya que es donde escribimos nuestro código. Pero no solo eso, también tenemos acceso a las funciones más utilizadas.

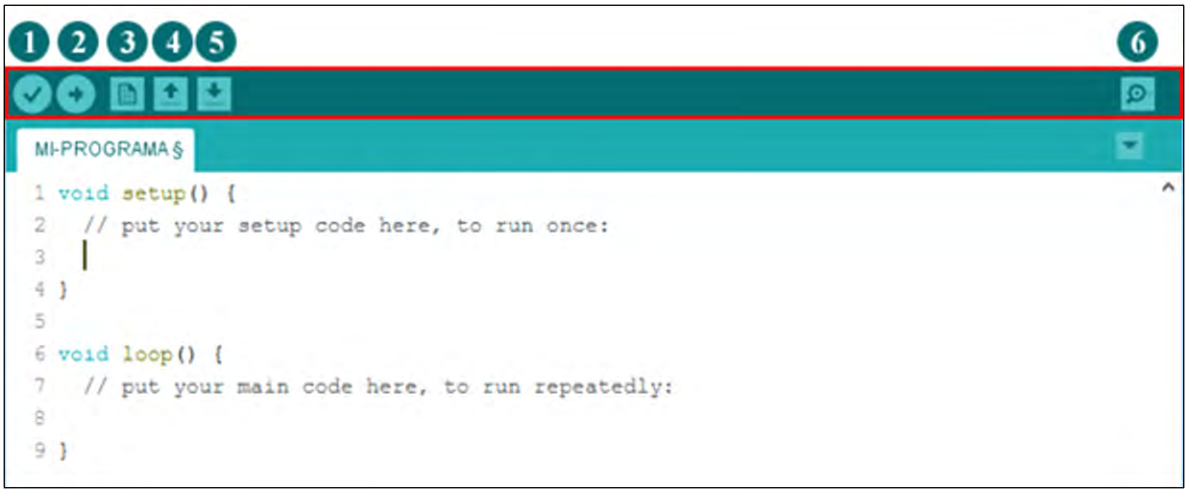
En la parte central encontramos el propio editor. Incluye el número de línea útil, por ejemplo, para detectar errores.

Figura 8. parte del editor con el código escrito del IDE



Fuente: Sistema Arduino

Figura 9. Iconos de accesos directos a las funciones más utilizadas.



Fuente: Sistema Arduino

Justo arriba del editor tenemos los accesos directos a las funciones más utilizadas.

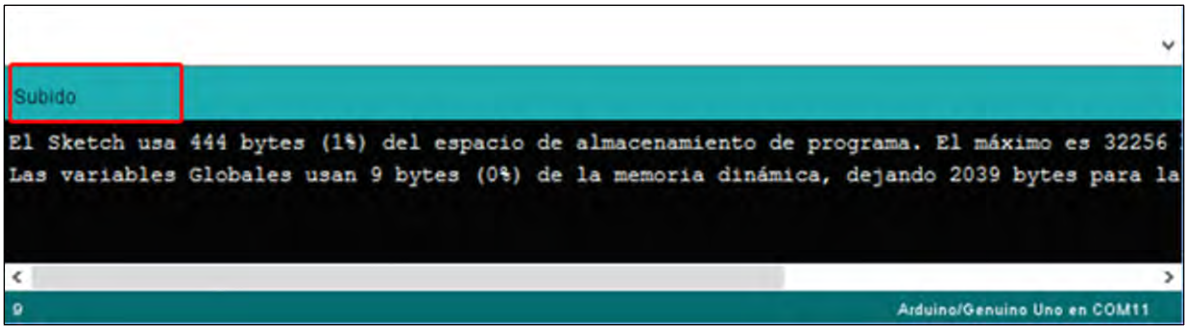
1) Verificar/Compilar: este botón verifica el código en busca de errores y lo compila. Cuando hablo de compilar me refiero a traducir el lenguaje de programación que entendemos los humanos en código máquina que entienden las máquinas.

2) Subir: el botón subir nos permite cargar o subir el código al microcontrolador a través del puerto serie USB.
3) Nuevo: sirve para crear un programa nuevo. Esto genera una nueva ventana donde escribir el código de ese nuevo programa.
4) Abrir: abre un programa que hayas guardado previamente en el disco duro.
5) Salvar: guarda el archivo en el disco duro. Es como la opción que hemos visto anteriormente.
6) Monitor serie: es una de las partes más importantes del IDE de Arduino. Sirve para mostrar información de la comunicación entre el ordenador y Arduino en las dos direcciones.

Todos estos accesos directos tienen su correspondencia en el menú de opciones y también tienen su atajo de teclado.

El área de mensajes:

Figura 10. Área en el cuál muestra el mensaje cuando se está realizando alguna tarea



Fuente: Sistema Arduino

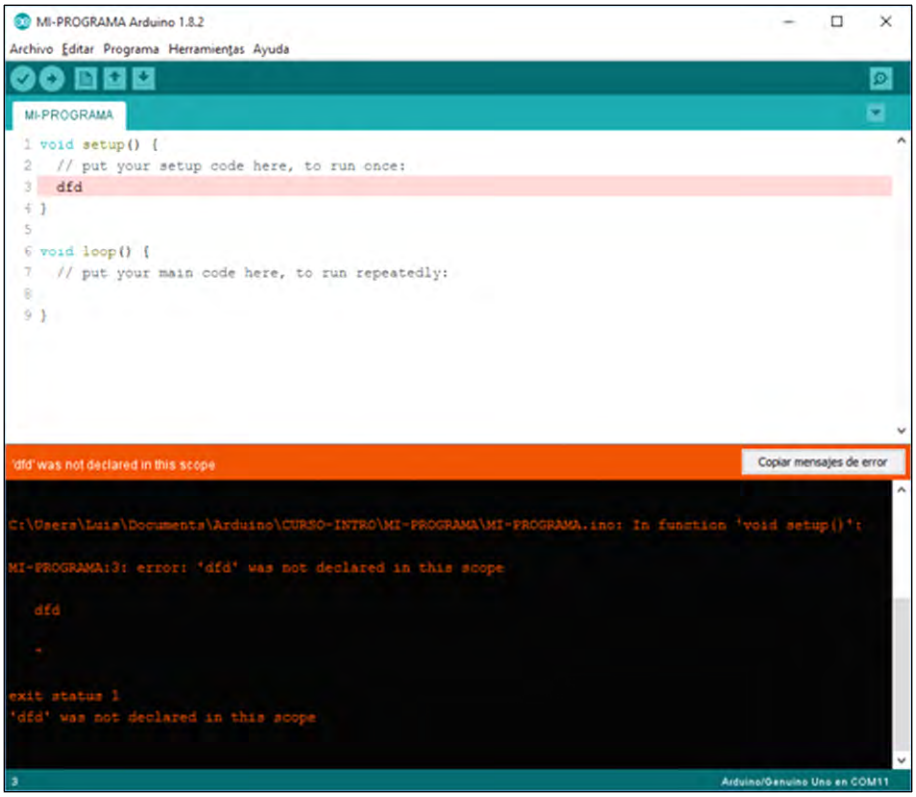
En esta área de mensajes se muestra la última acción que has realizado. También muestra mensajes cuando se está realizando alguna tarea como subiendo un programa a la placa.

La consola:

La consola nos va a dar información muy valiosa. Nos puede dar información sobre una acción concreta, por ejemplo, los datos tras subir un programa a la placa. Pero lo más importante, nos informa si hay algún error. (Crespo, 2017)

Figura 11.

Figura 12. El área negra llamada consola nos da información en casa de error

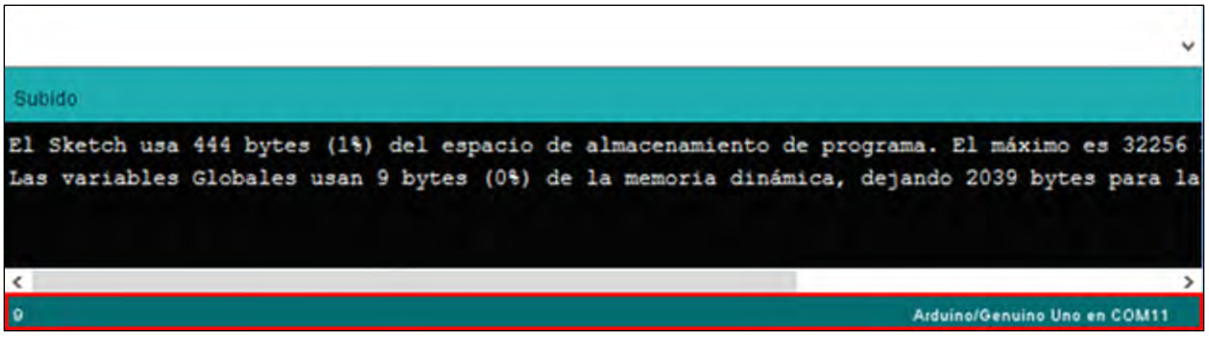


Fuente: Sistema Arduino

Otras partes importantes del IDE de Arduino

Una de las áreas donde podemos encontrar información muy valiosa es justo abajo del todo. Se pueden ver dos áreas de texto.

Figura 13. Paste inferior de la consola



Fuente: Sistema Arduino

En la parte izquierda nos informa del número de línea donde está situado el cursor. En la parte de la derecha tenemos un resumen de la placa que tenemos seleccionada y el puerto serie que estamos utilizando. (Hernández)

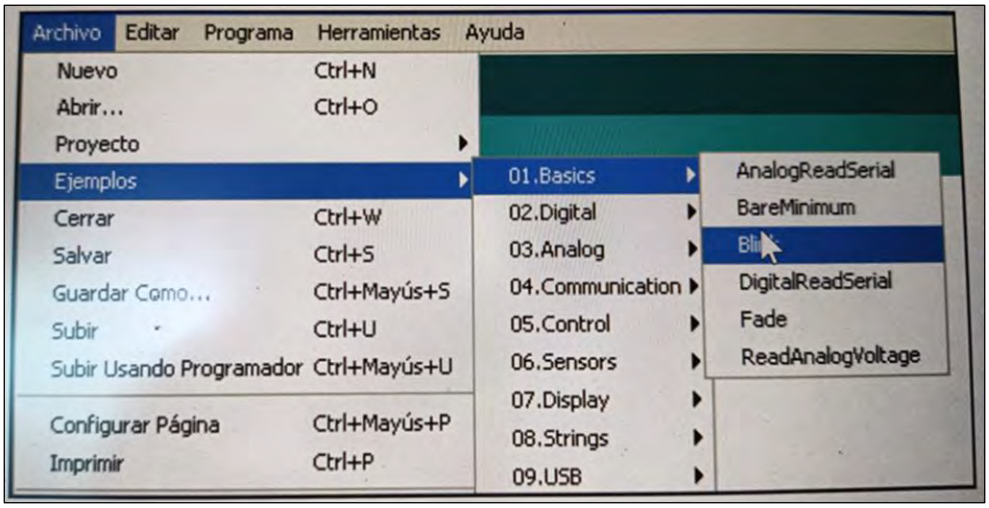
D. Comunicación con Arduino.

Ahora que ya ha instalado el IDE de Arduino y está seguro de que su ordenador puede Comunicarse con la placa, es el momento de comprobar si puede cargar un programa en Arduino.

1. Hacer doble Click sobre la aplicación de Arduino para abrirla. Si el entorno IDE está en un lenguaje que no es el suyo, puede cambiarlo al seleccionar el menú “Archivo” y escoger “Preferencias”. En la ventana que se abre y dentro del “Editor de idioma” escoger el lenguaje que desee. Reiniciar el programa para que tenga efecto. Buscar “the environment page” dentro de esta página para obtener más información: arduino.cc/ide.

2. Navegar para abrir un sketch de ejemplo que hace que el diodo LED de la placa Arduino parpadee, la palabra “sketch” es como se llama a los programas en Arduino. Seleccionar el menú “Archivo” escoger “Ejemplos” a continuación “01Basics” y por último “Blink”.

Figura 14. Abrir el programa “sketch” a 01 basic.blink

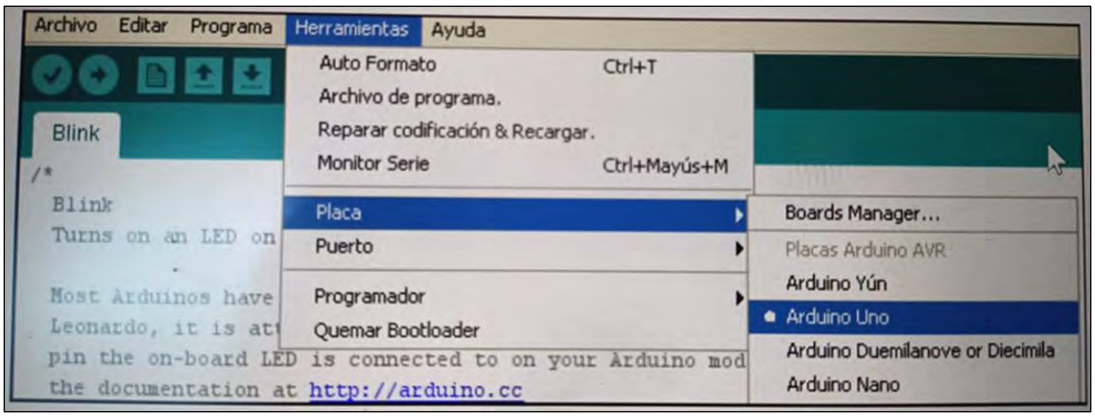


Fuente: Sistema Arduino

3. Debería de abrirse una nueva ventana con un texto dentro. Dejar la ventana como está por ahora, y seleccionar su placa Arduino desde:

Herramientas →placa.

Figura 15. Nueva ventana del sketch para seleccionar el tipo de placa



Fuente: Sistema Arduino

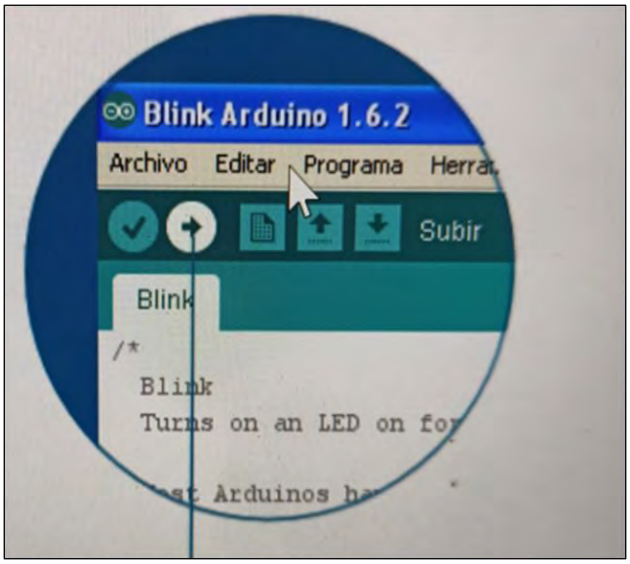
4. Escoger el puerto serie en donde su placa Arduino está conectada desde el menú de: Herramientas > Puerto

a) En Windows: Probablemente aparecerá con el número más alto de puerto COM. No pasará nada si se equivoca al escoger este número de puerto, simplemente no funcionará.

b) En Mac: Deberá aparecer algo parecido a /dev/tty.usbmoderm en este sistema. Por lo general aparecen dos, seleccionar uno cualquiera de ellos.

5. Para cargar el Sketch que hace que el diodo LED parpadee a su Arduino, presionar el botón Subir en la esquina superior izquierda de la ventana.

Figura 16. Ejemplo de cómo Cargar el sketch

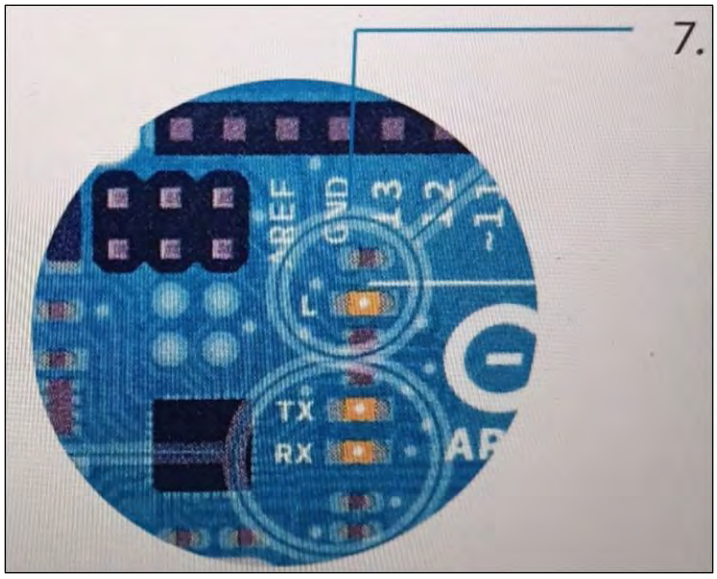


Fuente: Sistema Arduino

6. En la barra nos indicando el progreso de carga del sketch cerca de la esquina inferior derecha del IDE de Arduino, y los diodos LED de la placa Arduino con las etiquetas TX y RX estarán parpadeando en el momento de la carga. Si la carga se ha realizado correctamente, el IDE mostrará el mensaje SUBIDO en la esquina inferior izquierda.

7. A los pocos segundos de completar la carga del Sketch, debe observar como el diodo amarillo, con la etiqueta L cerca, comienza a parpadear. Ver imagen 1.16. (Crespo, 2017)

Figura 17. Muestra el parpadeo del diodo



.

Fuente: Sistema Arduino

E. Fundamentos de la placa de Arduino

Hasta ahora he hablado de las dos áreas que hay conocer para aprender Arduino: la programación y la electrónica. Hemos visto una introducción al IDE de Arduino que no es más que una herramienta para programar.

Más adelante veremos los fundamentos de Arduino. Hay multitud de placas de Arduino. Hay originales, copias y un amplio abanico de modelos dependiendo de cuál es su funcionalidad. No es lo mismo comprar una placa con conexión WiFi que sin ella. Por ejemplo, Arduino MKR1000 puede conectarse a Internet a través de la WiFi y transmitir datos.

Un Arduino UNO o un Arduino MEGA no pueden por si solos. Necesitan algún tipo de shield ya sea ethernet o WiFi.

Salvo estas características especiales de cada placa, el 99% del código te va a servir para cualquier placa. Por ejemplo, el acceso a los pines se hace en todas las placas de igual manera. Esa es la magia de Arduino. Un solo lenguaje y un solo IDE para dominar a todos.

En este caso estudiaremos con un Arduino UNO, sobre todo para desarrollar un Estudio Monográfico de Sensores de Nueva Generación para el sistema Arduino Uno y Node MCU.

Dentro de todos los modelos que podemos encontrar en la tienda oficial, el recomendado para aprender Arduino es el modelo Arduino UNO.

Se trata del buque insignia de la marca, el más famoso y el más vendido. Muchos otros modelos se han construido a partir de este.

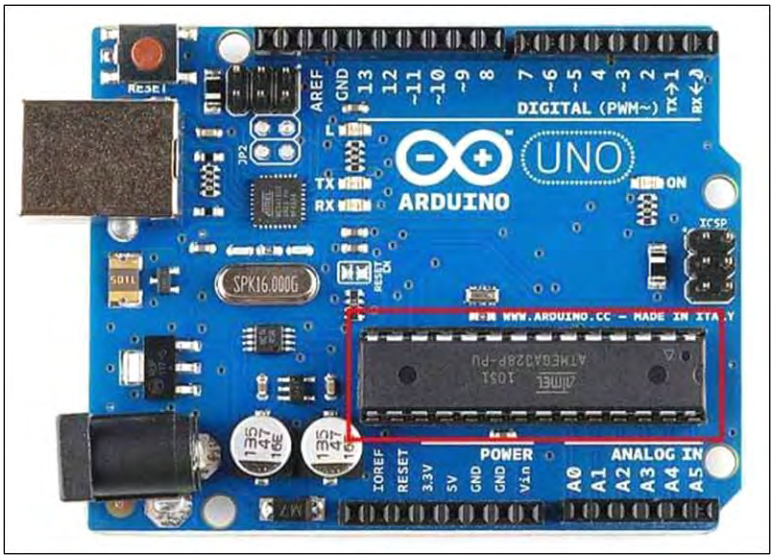
Figura 18. Modelo Arduino Uno

Fuente: Sistema Arduino

Una vez que te inicias con una placa Arduino, es muy sencillo utilizar otros modelos de placa e incluso de otras marcas. Cuando hablo de placas, lo realmente importante y sobre lo que todo gira es el microcontrolador que tienen integradas esas placas.

La realidad es que Arduino está creado con un único objetivo: facilitarnos la programación de un microcontrolador. Lo primero es identificarlo dentro de la placa. Si miras un Arduino de cerca, verás una cucaracha o pastilla negra donde pone ATMEL. Eso es el microcontrolador.

Figura 19. Microcontrolador ATMEL del modelo Arduino Uno



Fuente: Sistema Arduino

Los microcontroladores también se llaman MCU por sus siglas en inglés Microcontroller Unit. Diariamente se utilizan decenas de ellos en dispositivos electrónicos, electrodomésticos, coches, ordenadores, móviles, etc…

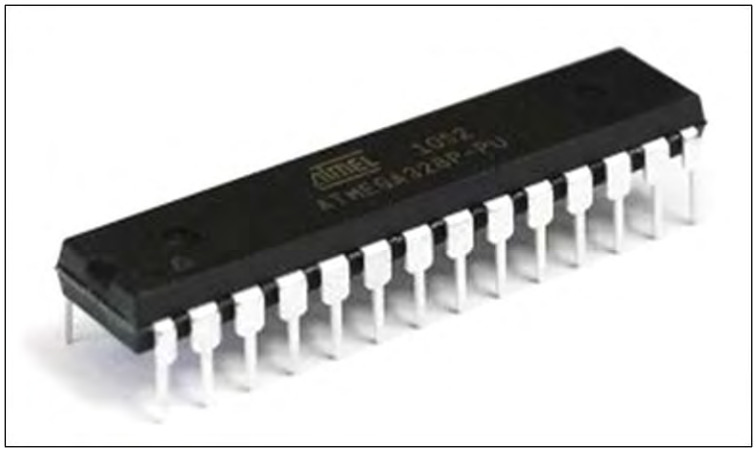
Es un circuito integrado programable, capaz de ejecutar las órdenes que están almacenadas en su memoria. Esto quiere decir que lo podemos programar.

Pero también existen los microprocesadores. Son los típicos procesadores que encontramos en los ordenadores, por ejemplo. Existe una diferencia principal entre una MCU o microcontrolador y un microprocesador, su utilidad.

Una MCU tiene como objetivo una tarea concreta. Por ejemplo, cerrar las puertas de un ascensor, captar la temperatura de un sensor, etc...

Sin embargo, un microprocesador es de propósito general. Puede hacer varias cosas a la vez. Recopilar información de los datos, enviar por email, mostrar en una pantalla, etc…

Figura 20. Concepto de placa de prototipo



Fuente: Sistema Arduino

Su nombre completo es ATMEGA328P-PU y es un microcontrolador de 8-bit**.** Esto quiere decir que solo puede hacer operaciones con números de 8-bit (números entre 0 y 255).

Ahora imagínate que tuvieras que programar este circuito integrado o chip, ¿cómo lo conectas al ordenador? ¿Cuáles son los pines digitales y analógicos? ¿Dónde está el pin de 5V y el de GND? 1.19- Imagen del microcontrolador ATMEL.

Hubo una persona que pensó que había que democratizar el uso de microcontroladores y creo Arduino. Él es David Cuartielles uno de los cofundadores de Arduino y el padre de la criatura.

Todo lo que rodea a la placa de Arduino está pensado para facilitarnos la programación y conexión con el microcontrolador.

La huella o forma en la que están dispuestos los pines, la conexión serie USB para programar y alimentar, el alimentador de baterías o pilas, cada componente está puesto en su sitio para que todo sea más fácil para nosotros.

A continuación, le presentamos las placas más populares, sus características y le damos a conocer los dos grandes bloques, las oficiales y las compatibles. (Crespo, 2017)

Tabla 1. Modelos de Arduino

Modelos de Arduino	
Arduino Uno	*Arduino Robot*
Arduino Leonardo	*Arduino Mini*
Arduino Due	*Arduino Nano*
Arduino Yún	*LilyPad Arduino Simple*
Arduino Tre (En Desarrollo)	*LilyPad Arduino SimpleSnap*
Arduino Zero (En venta en la tienda de EEUU)	*LilyPad Arduino*
Arduino Micro	*LilyPad Arduino USB*
Arduino Esplora	*Arduino Pro Mini*
Arduino Mega ADK	*Arduino Fio*
Arduino Ethernet	*Arduino Pro Basado en Arduino Leonardo (Atemega 32U4)*
Arduino Mega 2560	

Fuente: Elaboración propia

A continuación, tenemos una descripción de lo más importante de cada uno de las placas de sistema Arduino citados en la lista anteriormente mencionado, los cuales son lo más usados.

Arduino UNO

Esta placa es la más popular y la primera que salió al mercado en 2010. Utiliza un microcontrolador Atmel ATmega320 de 8bits a 16Mhz. Funciona a 5V y dispone de una memoria flash de 32Kb. Tiene 14 pines digitales y 6 analógicos. Puede parecer una placa muy limitada en cuanto a prestaciones, pero es suficiente para la mayoría de proyecto de tamaño pequeño o medio. Suele ser la placa que se utiliza para aprender a desarrollar en Arduino. (CEAC, 2018)

Arduino Mega

En contraposición a la UNO, esta es la placa más potente a 8 bits. Tiene el microcontrolador Atmega2560 con más memoria para el programa, más RAM y más pines que el resto de los modelos. (CEAC, 2018)

Arduino Due

Esta placa es la que tiene mayor capacidad de procesamiento con un microcontrolador Atmel SAM3X8E ARM Cortex-M3 de 32 bits trabajando a 84Mhz. La memoria flash es de 512Kb.Dispone de 54 pines digitales y 12 analógicos por lo que nos proporciona gran capacidad de conexionado. (CEAC, 2018)

Arduino Yun

Basada en el microcontrolador ATmega32u4 a 16Mhz y en un chip Atheros AR9331, es parecida a la UNO, pero con capacidades nativas para conexión Ethernet, Wifi, USB y microSD sin necesidad de agregar o comprar *shields*. Dispone de 20 pins digitales y 12 analógicos. Su memoria flash es de 32Kb. (CEAC, 2018)

Arduino Ethernet

Esta placa viene a cubrir las carencias de la UNO en cuanto a capacidades Ethernet. El microcontrolador es el ATmega328 que trabaja a 16Mhz. La memoria flash es de 32Kb y la misma distribución de pines que la UNO, es decir, 14 pines digitales y 6 analógicos. (CEAC, 2018)

Arduino Nano

La actual Arduino Nano monta un microcontrolador ATmega168 a 16Mhz. Es la placa más pequeña de todas por lo que necesita de un cable mini-USB y no posee conector de alimentación externa. Dispone de 14 pines digitales y 8 analógicos. Inicialmente disponía de una memoria flash de 32Kb que se ha reducido a 16Kb. (CEAC, 2018)

Arduino Leonardo

El Leonardo monta un microcontrolador ATmega32u4 de bajo consumo y que trabaja a 16Mhz. Su memoria flash es de 32Kb y dispone de 20 pines digitales y 12 analógicos. Utiliza una conexión mini-USB en vez de USB para reducir su tamaño respecto a una UNO.

El chipKIT Uno32 combina la compatibilidad con la popular plataforma de desarrollo para hardware de fuente abierta Arduino con el rendimiento del microcontrolador Microchip PIC32. El Uno32 tiene la misma configuración que la tarjeta Arduino Uno y es compatible con las shields Arduino.

Incluye una interface de puerto serial USB para conexión a IDE y puede ser energizada vía USB o un suministro de energía externa.

La tarjeta Uno32 tiene la ventaja de usar el poderoso microcontrolador PIC32MX320F128. Este microcontrolador incluye un procesador MIPS de 32- bits que corre a 80MHz, tiene una memoria de programa Flash de 128K y una SRAM de 16K. El Uno32 puede ser programado usando un ambiente basado en el IDE original de Arduino modificado para

soportar el PIC32. Adicionalmente, el Uno32 es totalmente compatible con el ambiente de desarrollo avanzado Microchip MPLAB y el sistema interno programador/debugging PICKit3.

Y hasta aquí algunas de las placas más populares, aunque hay más en el mercado sin hablar de las compatibles. El dilema que se nos puede plantear es cuál utilizar, en este caso, la más recomendable es el Arduino UNO R32. Tiene un coste sobre los $426.00 y nos permitirá realizar los primeros proyectos sin dificultad.

Nunca hay que perder el foco del objetivo que pretenden todas las placas de Arduino: programar un microcontrolador. Dicho esto, un Arduino Uno se puede utilizar para lo que quieras. Desde un simple dado electrónico hasta un robot, Arduino controlado por Bluetooth. El límite lo pone la imaginación.

Dentro de la gran diversidad de placas que hay en el mercado, tenemos que tener en cuenta que hay dos grandes bloques.

Por un lado, encontramos las placas oficiales que son las que produce la compañía Smart Project. Los diseños de estas han sido realizados por diferentes empresas.

Una placa es oficial porque llevan el logo de Arduino y están certificadas para trabajar con el entorno de desarrollo o IDE de Arduino.

Por otro lado, tenemos placas compatibles con Arduino. Estas placas no pueden estar registradas como Arduino. La única diferencia con las oficiales es que puede que estas no sean compatibles en todos los sentidos, es decir, a nivel de hardware o del entorno de desarrollo.

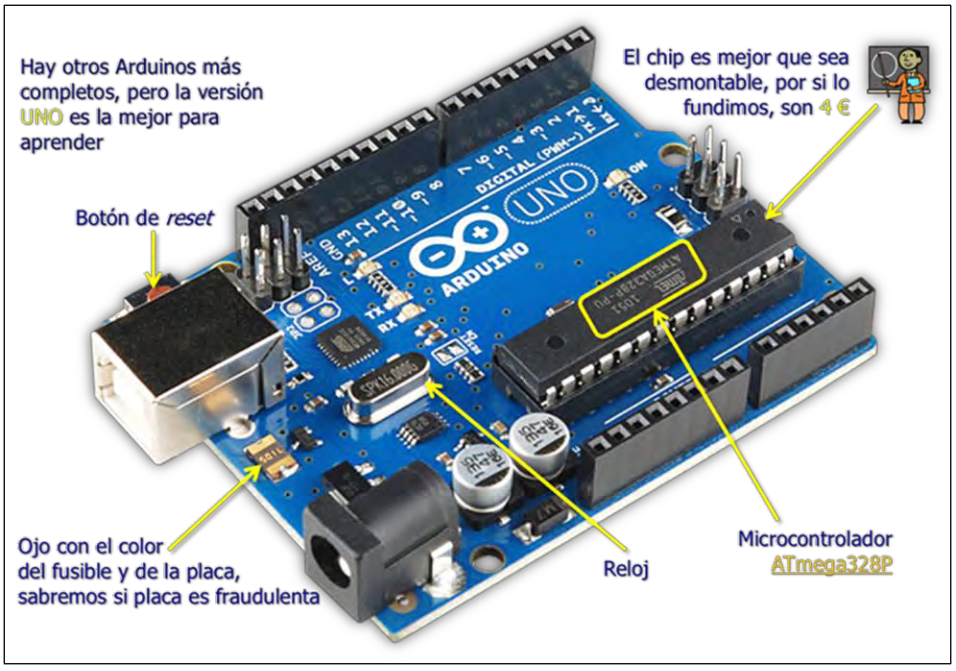
Características a tener en cuenta para seleccionar el tipo de Arduino.

- Antes de mencionar las diferentes placas existentes, debemos tener claro una serie de aspectos que determinarán qué placa necesitaremos.
- Se debe tomar en cuenta si los proyectos van a ser de códigos amplios o no, esto determinará la necesidad de memoria de la placa.

- También es importante saber si se requiere conectar muchos periféricos o no, para utilizar una placa con más o menos pines.
- Otro aspecto para considerar es si necesitamos o no conexión wiki, lo que determinará escoger una placa en concreto o bien añadirle un shields.
- El tamaño de esta también puede llegar a ser importante en función del uso que se le pueda dar.
- En algunos tipos de proyecto los valores de tensión LOW y HIGH pueden ser importantes. No todas las placas funcionan con los mismos umbrales para determinar si es un valor alto o bajo.
- Y, por último, creo que debe tenerse en cuenta también la tensión de alimentación que puede ser de 12V o 5V. (CEAC, 2018)

Placa Arduino uno.

Figura 21. Las partes que componen la Placa sistema Arduino Uno.



Fuente: Sistema Arduino

Esta plataforma permite crear diferentes tipos de microcontrolador de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso. Para entender mejor este concepto es necesario conocer que es un hardware libre y el software libre.

El hardware libre son los dispositivos cuyas especificaciones y diagramas son de acceso público, de manera que cualquiera puede replicarlos. Esto quiere decir que Arduino ofrece las bases para que cualquier otra persona o empresa pueda crear sus propias placas, pudiendo ser diferentes entre ellas, pero igualmente funcionales a partir de la misma base.

El software libre son los programas informáticos cuyo código es accesible, para que quien quiera pueda utilizarlo y modificarlo. Arduino ofrece la plataforma Arduino IDE (Entorno de Desarrollo Integrado), que es un lenguaje de programación propio, simplificado con el que cualquiera puede crear aplicaciones para las placas Arduino, de manera que se les puede dar todo tipo de utilidades. Esto permite acercar el mundo de la electrónica a personas con pocos conocimientos sobre microcontroladores o con pocos recursos económico, ya que las placas Arduino tienen todos los elementos necesarios para conectar periféricos a las entradas y salidas de un microcontrolador, y que puede ser programada tanto en Windows como Mac OS X y GNU/Linux.

La programación es directamente desde su IDE, con un simple cable USB, sin necesidad de programadores específicos. Esta sencillez, se consigue gracia a un bootloader, que viene precargado en el microcontrolador.

La preferencia del Arduino UNO dentro de la familia Arduino se debe a varios factores:

- Bajo precio: es una de las placas de desarrollo más económicas del mercado.
- Compatibilidad: la posición de sus cabezales hace que sea compatible con casi todos los shields y accesorios Arduino existentes.
- Presencia en Kits de desarrollo: la mayoría de los kits de Arduino se basan en esta placa.

- Fácil de utilizar: al igual que la mayoría de las placas Arduino es extremadamente simple implementar prototipos con ella.
- Robusta: es posible utilizarla sin mucha preocupación ya que en el peor de los casos se puede remplazar el microcontrolador (en los modelos que utilizan encapsulados DIP) y listo.

La evolución de las placas Arduino se divide en 2 etapas:

1.- Arduino Serial: primeras placas de Arduino. Utilizaban una interfaz RS232 para comunicarse con el ordenador.

Las placas derivadas del proyecto Arduino fueron conocidas como Arduino Serial y Arduino Serial v2.0. Se presentaron en sociedad en los años 2005 y 2006. El término "Serial" se debe a que para grabar el programa se utilizaba el protocolo serie RS-232 mediante un puerto serie.

2.- Arduino USB: Estas placas sustituyen la interfaz RS232 con un puerto USB para simplificar el proceso de grabado y comunicación con el ordenador.

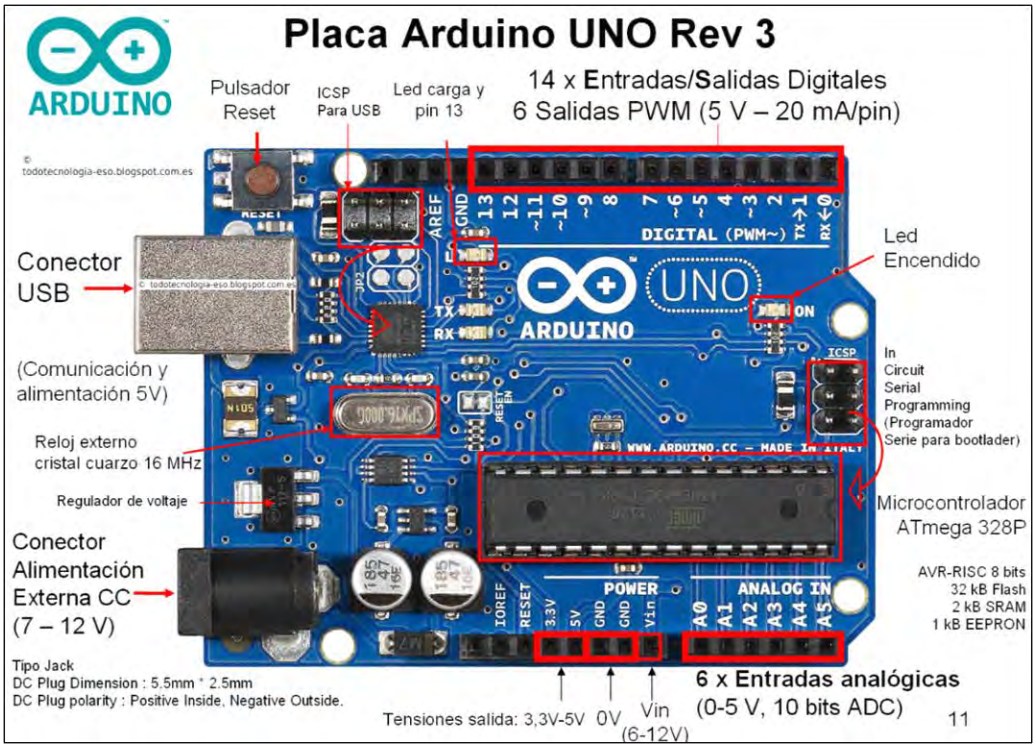
Los ordenadores modernos no traen ya estos puertos (también llamados puertos COM) es necesario utilizar un adaptador USB-RS232 para programar esta placa.

Como mencionamos anteriormente La placa Arduino con la que se realizara el estudio monográfico es el modelo Arduino Uno.

La placa Arduino Uno es la base del gran imperio del Arduino, es la placa más popular y accesible de Arduino. Se basa en el chip ATmega328. Arduino Uno es la opción más adecuada para empezar a trabajar con la plataforma, se puede conseguir muy fácilmente y a muy bajo coste.

A continuación, le presentamos las especificaciones, características y dispositivos de la placa Arduino Uno, los requisitos de alimentación y la capacidad de conectar dispositivos externos.

Figura 22. Descripción de las partes e Arduino Uno



Fuente: Sistema Arduino.

Arduino UNO es una placa de desarrollo basada en el microcontrolador ATmega328P. Es una de las más conocidas y usadas de la extensa familia de placas Arduino, siendo una mejora de un diseño anterior Arduino Duemilanove que mantiene el 100% de compatibilidad. Su nombre “UNO”, fue seleccionado para marcar el lanzamiento de la versión 1.0 del software Arduino IDE. (CEAC, 2018)

Características de Arduino uno.

Tabla 2. Características Arduino Uno

Arduino Uno.	
Microcontrolador ATmega328	*Corriente máxima de una salida 40 mA*
Tensión de funcionamiento 5v	*Corriente máxima de salida 3.3V 50 Ma*
Tensión de alimentación (recomendada) 7-12v	*Memoria flash 32KB (ATmega328) de los cuales 0,5KB son utilizados por el cargador*
Tensión de alimentación (límite) 6-20v	*SRAM 2KB (ATmega328)*
E/S digitales 14 (de los cuales 6 pueden utilizarse como salidas PWM)	*EEPROM 1 KB (ATmega328)*

Entradas analógicas 6	*Frecuencia de reloj 16 MHz.*
Clones de Arduino Uno R3	*Imágenes de clones Arduino Uno compatibles.*

Fuente: Elaboración propia

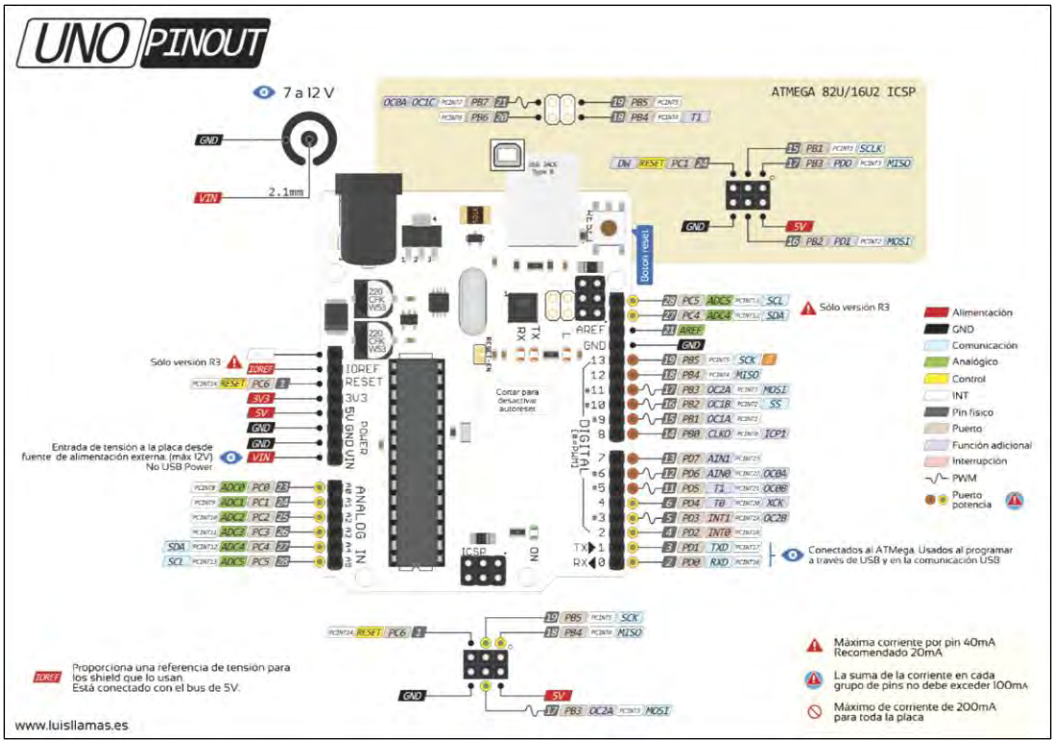
Esquema placa Arduino uno.

El esquema de patillaje de un dispositivo electrónico, o pinout, es uno de los documentos de referencia más útiles y que más frecuentemente consultaremos a la hora de realizar un montaje. El caso de Arduino no es una excepción, más aun teniendo en cuenta la gran cantidad de pines y modelos de placas disponibles.

A medida que la complejidad de los circuitos aumenta y empieza a emplear funciones más complejas, como comunicación entre dispositivos o interruptores, resulta más conveniente disponer a mano de estos esquemas, para poder consultar rápidamente los datos de cada pin en las distintas placas en uso.

Esquema de pinout de la placa Arduino Uno, que serán documentos de consulta que emplean frecuentemente en el resto de entradas del blog. (Lamas, 2018)

Figura 23. Pinout diagrama de la placa Arduino UNO



Fuente: Sistema Arduino

El esquema Arduino uno cuenta con tres bloques principales

- La etapa de alimentación: encargada de que los componentes de la placa reciban los voltajes adecuados.
- La interfaz de programación: permite que el microcontrolador sea programado desde el ordenador.
- Microcontrolador con “sistema mínimo”: componentes indispensables para que el microcontrolador funcione correctamente.

Etapa de alimentación Arduino Uno.

El primer elemento de esta etapa es el conector tipo Jack imagen1.17. Este se conecta al regulador de voltaje (IC2) mediante el diodo (D1). Dicho diodo ofrece protección a toda la placa en caso de que la polaridad de la fuente sea invertida.

El regulador IC2 ofrece a su salida 5 voltios estables. Ese voltaje es utilizado por el microcontrolador y la interfaz de programación. Los condensadores que acompañan al regulador son utilizados para mejorar la estabilidad de la fuente y evitar variaciones de voltaje.

Por último se ha incluido un LED (POWERLED) con una resistencia de 220Ohmios. Este LED indica que la placa está correctamente alimentada.

Interfaz de programación Arduino Uno.

Como interfaz de programación se utiliza un conector DE-9. Este conector debe ser conectado a un puerto serie del ordenador o a un adaptador USB-Serie si la computadora no dispone de uno.

Estos puertos implementan el protocolo RS232 que es similar al utilizado por el microcontrolador. Su principal diferencia reside en los voltajes empleados para los niveles lógicos.

La siguiente tabla muestra los voltajes empleados para representar los niveles lógicos en ambos casos.

Nivel Lógico	*Seri TTL (Microcontrolador)*	*Serie RS-232(E-9)*
1	+5v	-3; -15v
0	0v	+3v; +15v

Las formas de onda del protocolo TTL y RS-232(DE-9), son diferentes, esto provoca que no sea posible vincular directamente los pines del conector a los del microcontrolador, necesitando un conversor de niveles.

Esta función la realizan los transistores, diodos, condensadores y resistencias que acompañan al conector en el diagrama. Las etiquetas M8RXD y M8TXD representan la

conexión de los pines RXD y TXD del microcontrolador con la interfaz de programación. (Programarfácil, 2018)

Microcontrolador con “sistema mínimo”

En este bloque se encuentra el eje principal de la placa: el microcontrolador. En esta placa se utiliza un ATmega8. Este cuenta con una memoria de programa de 8 KByte, una memoria de datos de 1 KByte y 512 Bytes de memoria EEPROM. Este dato es el más significativo, sin embargo, se debe consultar la hoja de datos y comprobar qué otras características poseen este microcontrolador.

Ahora presentamos el circuito que lo rodea:

En primer lugar está el pin RESET. Este pin está conectado mediante una resistencia de 10 kΩ a los 5 V de la fuente y al botón S1 (este es el botón de reinicio).

Esta configuración se debe a que cuando el pin RESET es puesto a nivel bajo el microcontrolador es reiniciado (esto ocurre al presionar el botón).

Para salir del reinicio es necesario ponerlo a 5 voltios nuevamente (gracias a la resistencia R1 esto ocurre al liberar el botón).

Los pines XTAL1 y XTAL2 están conectados al circuito oscilador formado por el cristal Q1 y los capacitores C2 y C3. Este circuito es el que permite la ejecución del programa almacenado dentro del microcontrolador, una vez este es energizado.

Cabezales de pines

Esta placa cuenta con 5 cabezales de pines: como se muestra en la imagen 1.17.

J1: este cabezal contiene los primeros pines digitales (D0…D7).

J2: es el cabezal de entradas analógicas.

J3: contiene el resto de los pines digitales (D8…D13), además, cuenta con un pin conectado a tierra (GND) y el pin de referencia analógico (AREF).

POWER: este cabezal tiene dos pines de tierra (GND), uno de 5 voltios (5V) y otro conectado a la entrada del regulador (VIN). Este último puede ser utilizado para alimentar la placa en caso de que el conector Jack no sea empleado.

ICSP: conector ICSP que permite programar el microcontrolador utilizando un programador externo.

Entre el pin PB5 (pin digital 13) del microcontrolador y su cabecera se ha colocado un resistor de 1KOhmio. Esto permite que un led pueda ser conectado directamente entre este pin y tierra sin necesidad de resistor externo.

Adaptador USB-RS232: la sustitución del puerto RS-232 por un puerto USB.

Teniendo en cuenta que se necesita una comunicación serie utilizando un puerto USB, se empleó un chip FTDI FT232BM. Este chip es capaz de emular un puerto Serie utilizando un puerto USB

A diferencia de la mayoría de los adaptadores USB-Serie, este chip utiliza niveles lógicos TTL para la comunicación, por lo tanto, se puede conectar directamente al microcontrolador.

FTDI FT232BM

Si inspeccionamos su hoja de datos encontramos que es muy versátil. Pero su principal finalidad es actuar como un convertidor de USB a RS232, RS422 /RS485. Además, es compatible con la mayoría de los sistemas operativos, convirtiéndolo en una muy buena opción.

EL CHIP CUENTA CON 32 PINES:

Estos se catalogan en 6 grupos:

- Grupo 1 (Interfaz USB): está compuesto por los pines USBDP y USBDM. Estos se conectan a los pines D+ y D- del puerto USB respectivamente.

- Grupo 2. (Alimentación), agrupa los pines relacionados con la alimentación.

Veamos los más importantes:

VCCIO: Empleado para indicar el voltaje a utilizar por las señales de la interfaz UART. (En caso de necesitar una interfaz a 5 voltios puede ser unido a VCC)

GND: Pines de conexión a tierra

3V3OUT: Ofrece un voltaje estable de 3.3 voltios (No puede suministrar más de 5 mili-amperios).

VCC: Voltaje de Alimentación del chip (5 voltios típicamente).

- Grupo 3 (Señales Misceláneas), provee pines para:

Reinicio del chip.

Agregar leds indicadores de transmisión y recepción de datos.

Conectar un cristal oscilador o resonador. Y para activar el modo de prueba.

- Grupo 4 (Interfaz UART): Agrupa los pines con todas las señales presentes en un conector RS-232 pero utilizando como nivel lógico alto el voltaje aplicado al pin VCCIO.
- Grupo 5 (Interfaz EEPROM): Permite conectar una memoria EEPROM con información (VID, PID, Número de Serie, Descripción, etc.) que se le da al ordenador en el momento de conexión USB.
- Grupo 6 (Control de Alimentación): Permite conocer el modo de operación del chip.

Todos los pines de alimentación cuentan con un condensador conectado a tierra para evitar variaciones de voltaje. Un resonador de 6 MHz es conectado a los pines XTIN y XTOUT.

En el esquema original también aparece un dispositivo extra y un conector DE-9. Pero en nuestro caso no es necesario, ya que este es utilizado para convertir a los niveles RS-232.

Como has podido comprobar, con unos pocos componentes es posible obtener un adaptador USB-Serie mucho más simple y robusto que el esquema empleado para el Arduino Serial. (Programarfácil, 2018)

Microcontrolador ATmega328 SMD vs DIP

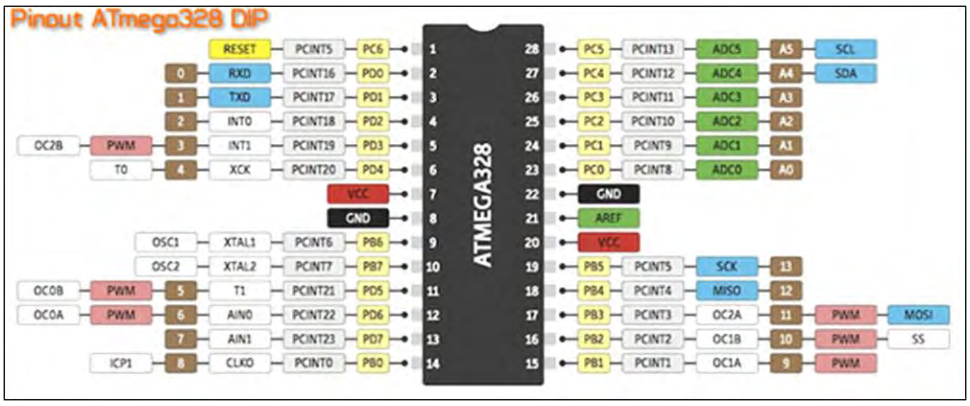
El microcontrolador ATmega328 del Arduino Uno puede venir montado en la placa de dos formas: soldado a la placa (SMD) y desmontable (DIP). Ambas versiones son muy similares, la única diferencia significativa además del aspecto es que el modelo desmontable se puede reemplazar y el que viene soldado no. (E., 2021)

Figura 24. Placa Arduino con el microcontrolador ATmega328 en sus dos versiones

Fuente: Sistema Arduino

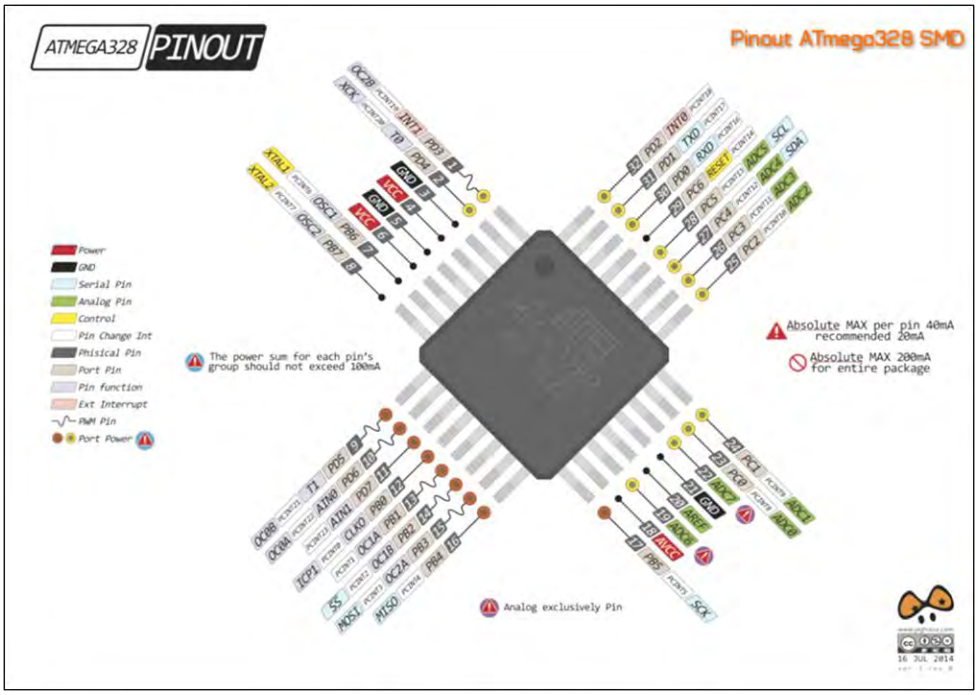
Pinout del Microcontrolador ATmega328.

Pinout Atmega328 DIP



Fuente: Sistema Arduino

Figura 25. Pinout ATmega328 SMD.

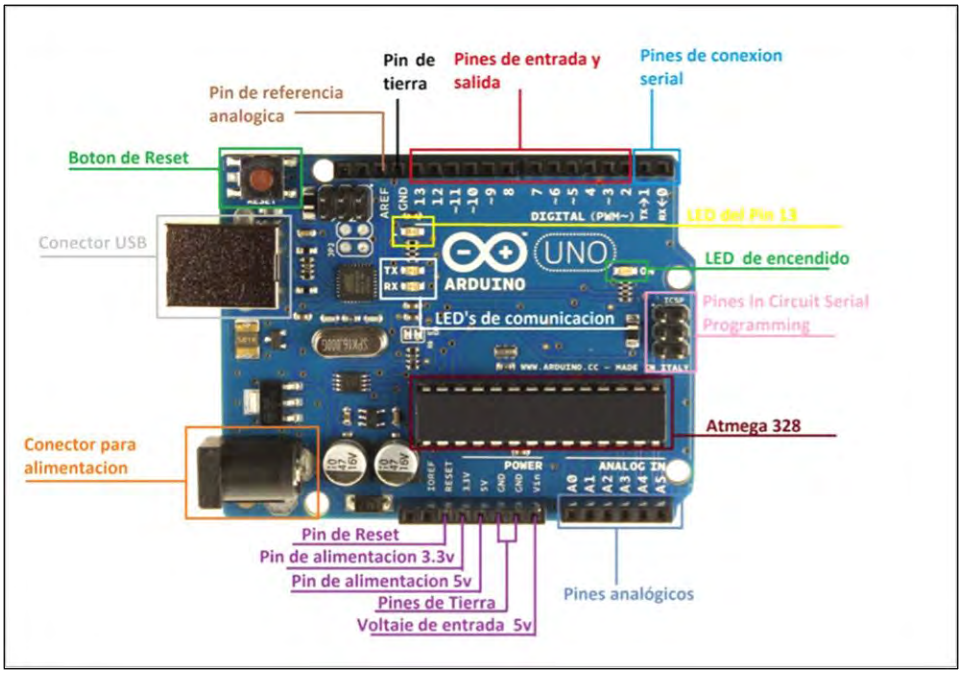


Fuente: Sistema Arduino

Descripción de los pines de Arduino

Los pines de Arduino se utilizan para la conexión de dispositivos externos y pueden funcionar tanto en los modos de entrada como de salida (OUTPUT). Una resistencia incorporada de 20-50 kOhm puede conectarse a cada entrada ejecutando el comando pinMode () en el modo INPUT_PULLUP. La corriente admisible en cada una de las salidas es de 20 mA, no más de 40 mA en pico. Para facilitar la operación, algunos pines combinan varias funciones.

Figura 26. Pines Arduino UNO.



Fuente: Sistema Arduino

Arduino Uno es una gran opción para crear tus primeros dispositivos inteligentes. 14 pines digitales y 6 pines analógicos te permitirán conectar una gran variedad de sensores, Leds, motores y otros dispositivos externos. El conector USB te ayudará a conectarte al ordenador para instalar el programa sin necesidad de dispositivos externos adicionales. El estabilizador incorporado te permite usar una variedad de baterías con un amplio rango de voltaje, de 6-7 a 12-14 V. En el Arduino Uno se implementó muy convenientemente el trabajo con protocolos populares: UART, SPI, I2C. (Arduino P. c., 2022)

Pines digitales de la placa Uno.

Los pines con números del 0 al 13 son digitales. Esto significa que sólo puede leer y recibir dos tipos de señales: ALTA y BAJA. PWM también le permite usar puertos digitales para controlar la potencia de los dispositivos conectados.

Tabla 3. Tabla de pines digitales Arduino Uno

PIN Arduino	*Código en Pinout*	*Propósito especial*	*PWM*
Pin digital 0	0	Rx	
Pin digital 1	1	Tx	
Pin digital 2	2	Entrada de interrupción	
Pin digital 3	3	Entrada de interrupción	PWM
Pin digital 4	4		
Pin digital 5	5		PWM
Pin digital 6	6		PWM
Pin digital 7	7		
Pin digital 8	8		
Pin digital 9	9		PWM
Pin digital 10	10	SPI (SS)	PWM
Pin digital 11	11	SPI(MOSI)	PWM
Pin digital 12	12	SPI(MISO)	
Pin digital 13	13	SPI(SCK) También conecta un LED incorporando la salida(disponible en la mayoría de las placas Arduino)	

Fuente: Elaboración propia

- Los pines 0 y 1 son contactos UART (RX y TX respectivamente).
- Pines 10 a 13 – Contactos SPI (SS, MOSI, MISO y SCK, respectivamente)

- Los pines A4 y A5 son los contactos I2C (SDA y SCL, respectivamente). (Arduino P. c., 2022)

Pines analógicos Arduino Uno

Los pines analógicos Arduino Uno están diseñados para conectar dispositivos analógicos y son entradas para el convertidor analógico-digital (ADC) incorporado, que es un Arduino uno de diez dígitos. (Arduino P. c., 2022)

Tabla 4. Tablas pines análogos Arduino Uno.

PIN ARDUINO	*CODIGO EN PINOUT*	*PROSITO ESPECIAL*
Pin Analógico A0	A0 o 23	
Pin Analógico A1	A1 o 24	
Pin Analógico A2	A2 o 25	
Pin Analógico A3	A3 o 26	
Pin Analógico A4	A4 o 27	12C (SCA)
Pin Analógico A5	A5 o 28	12C (SCL)

Fuente: Elaboración propia

Pines adicionales de la placa Arduino uno

AREF – proporciona una tensión de referencia para el ADC incorporado. Puede ser controlado por analogReference ().

RESET – Una entrada de señal baja en esta entrada hará que la unidad se reinicie.

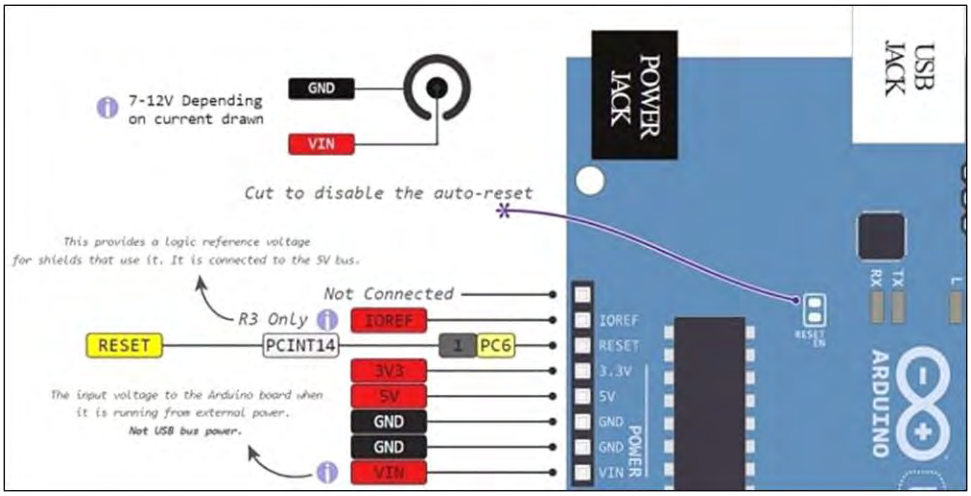
Opciones de alimentación de Arduino Uno

El voltaje de funcionamiento de la placa Arduino Uno es de 5 V. La tarjeta está equipada con un regulador de voltaje, por lo que la entrada puede ser alimentada desde diferentes fuentes y a través de dispositivos USB. La fuente de alimentación se selecciona automáticamente.

Hay 3 formas de alimentar el Arduino Uno:

- Barrel Jack – El Barrel Jack, o DC Power Jack puede ser usado para alimentar tu placa Arduino. El conector cilíndrico suele estar conectado a un adaptador de pared. La tarjeta puede ser alimentada por 5-20 voltios, pero el fabricante recomienda mantenerla entre 7-12 voltios. Por encima de 12 voltios, los reguladores podrían sobrecalentarse, y por debajo de 7 voltios, podrían no ser suficientes.
- VIN Pin – Este pin se utiliza para alimentar la placa Arduino Uno utilizando una fuente de alimentación externa. El voltaje debe estar dentro del rango mencionado anteriormente.
- Cable USB – Cuando se conecta al ordenador, proporciona 5 voltios a 500mA.

Figura 27. Esquemas formas de alimentación de los pines placa Arduino UNO



Fuente: Sistema Arduino

Podemos mencionar que hay un diodo de protección de 1 amperio entre el positivo de la patilla del barril y la patilla VIN.

La fuente de alimentación que usemos determinará la potencia disponible para nuestro circuito, ya que esta también alimentará la MCU, periféricos, reguladores y componentes que conectemos a nuestra placa.

Pines de alimentación

- 5V – este pin se suministra con 5V de Arduino y se puede utilizar para alimentar dispositivos externos.
- 3.3V – este pin es suministrado con voltaje de 3.3V desde el estabilizador interno
- GND es Tierra.
- VIN – pin para la alimentación de tensión externa.
- IREF – pin para informar a los dispositivos externos sobre la tensión de funcionamiento de la tarjeta.

(maker, s.f.)

Memoria Arduino Uno

De forma predeterminada, la tarjeta Uno admite tres tipos de memoria:

- Flash – 32 kB de memoria. Este es el principal almacenamiento de comandos. Cuando limpias el controlador con tu boceto, está escrito aquí. 2kB de esta reserva de memoria se asigna al bootloader, un programa que inicializa el sistema, arranca a través de USB y ejecuta el sketch.
- Memoria SRAM rápida de 2 kB. Las variables y objetos creados durante la operación del programa se almacenan aquí por defecto. Esta memoria depende de la energía y, por supuesto, todos los datos se borrarán cuando se desconecte la alimentación.
- Memoria no volátil (EEPROM) 1 kB. Aquí puede almacenar datos que no se borrarán cuando el controlador esté apagado. Sin embargo, el procedimiento de escritura y lectura de la EEPROM requiere el uso de una librería adicional, que está disponible en el IDE de Arduino por defecto. También es importante tener en cuenta la limitación de los ciclos de sobreescritura inherentes a la tecnología EEPROM.

Algunas modificaciones de la tarjeta Arduino Uno pueden admitir memoria con valores más altos que la estándar. Pero debe entender que necesitará librerías adicionales para trabajar con ellas también.

Programación de la placa Arduino Uno

Para escribir programas (sketches) para el controlador Arduino, necesitas instalar un entorno de programación. La opción más fácil es instalar un IDE de Arduino gratuito, después de instalar el IDE, debe asegurarse de que se ha seleccionado la placa correcta. Para ello, selecciona la placa Arduino en el menú «Herramientas» y en el submenú «Placa» del IDE de Arduino, luego de seleccionar la placa, los parámetros de montaje del proyecto se modificarán automáticamente y el modelo final se compilará en un formato que soporte la placa. Conectando el controlador al ordenador a través del puerto USB, puede instalar el programa con un solo Click utilizando el comando «Descargar».

El programa en sí mismo suele ser un ciclo sin fin en el que los pines con los sensores conectados se interrogan regularmente y la acción de control sobre los dispositivos externos (se encienden o se apagan) se realiza mediante comandos especiales. El compilador de Arduino tiene la capacidad de utilizar librerías ya hechas, tanto las que vienen incluidas en el IDE, como las que se tiene disponibles en sitios y foros.

El programa escrito y compilado se descarga a través de una conexión USB (UART-Serial). En el lado del controlador, el bootloader es responsable de este proceso.

Descargar el ide.

Descargar el entorno integrado de desarrollo (IDE) de Arduino para poder programar la placa de Arduino le permite escribir los programas y cargarlos a su placa Arduino.

arduino.cc/download.

Coloque la placa Arduino y el cable USB cerca de su ordenador. Todavía no los conecte. Seguir las instrucciones que se muestran a continuación para realizar la instalación para Windows, Mac OS X o Linux.

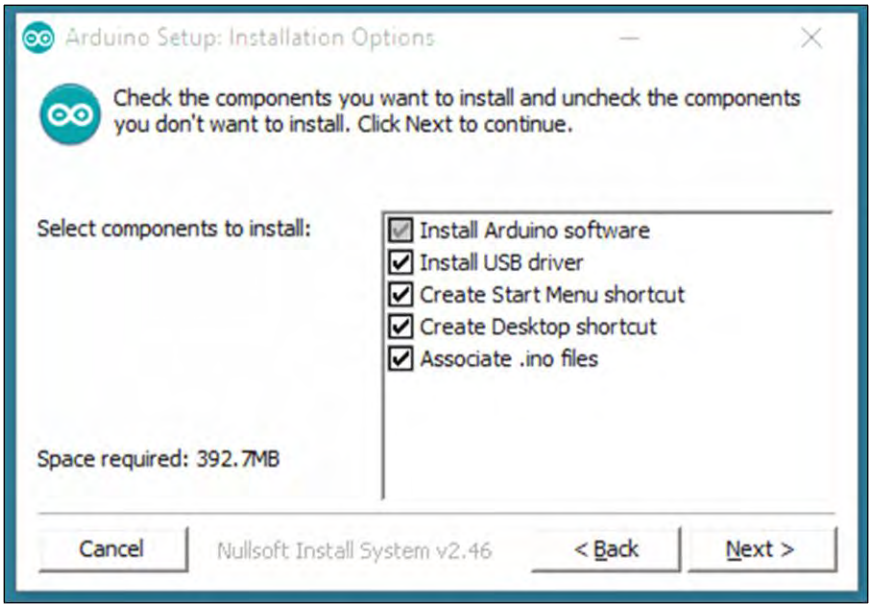
Breve explicación de cómo instalar el software Arduino (IDE) en la máquina con Windows.

Descargar e instalación el software Arduino (IDE)

Puede elegir entre el instalador (.exe) y los paquetes Zip. Le sugerimos que use el primero que instala directamente todo lo que necesita para usar el software Arduino (IDE), incluidos los controladores. Con el paquete Zip, se debe instalar los controladores manualmente. El archivo Zip también es útil si desea crear una instalación portátil .

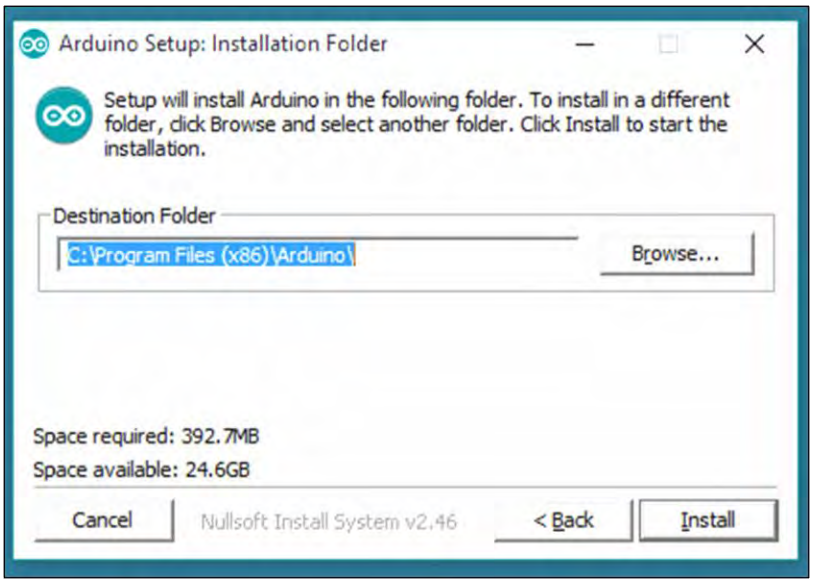
Cuando finalice la descarga, continúe con la instalación y permita el proceso de instalación del controlador cuando reciba una advertencia del sistema operativo.

Figura 28. Recuadro de dialogo para seleccionar los componentes a instalar.



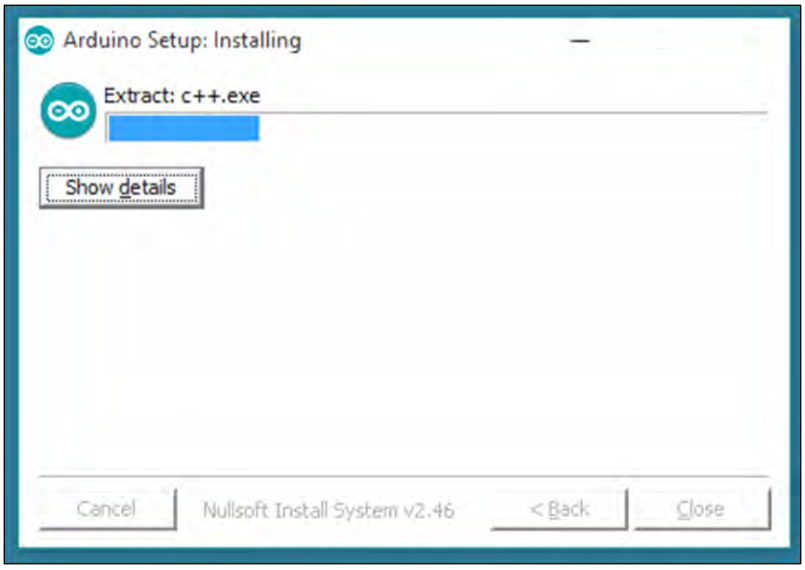
Fuente: Sistema Arduino

Figura 29. Recuadro de dialogo para elegir el directorio de instalación.



Fuente: Sistema Arduino

Figura 30. Recuadro de dialogo de Instalación en curso



Fuente: Sistema Arduino

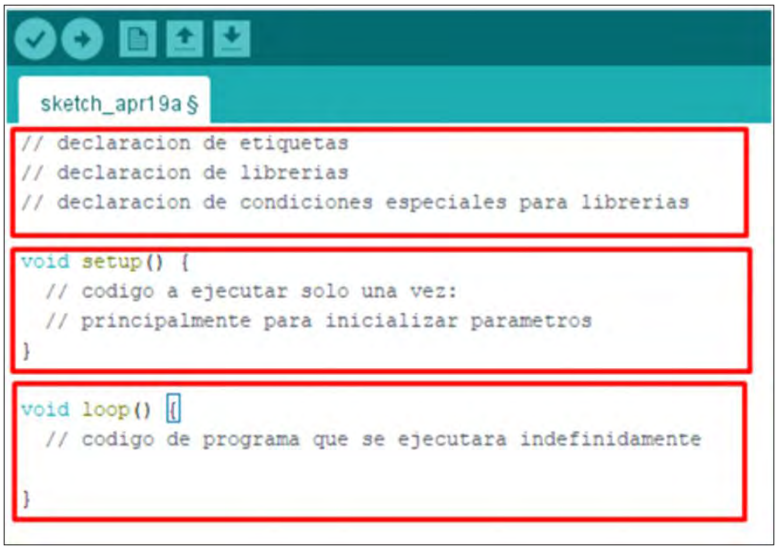
El proceso extraerá e instalará todos los archivos necesarios para ejecutar correctamente el software Arduino (IDE).De esta manera queda instalado el entorno de Desarrollo (IDE) de Arduino en la computadora. Como se mencionó antes también se puede trabajar en LÍNUX, y EN MAC OS X, para instalar el programa puede visitar la página web para ver como se hace dicha instalación. Arduino.cc/Linux

Cómo usar y administrar librerías

Probablemente a medida que avanzamos con el uso de esta plataforma, se requiere más herramientas y recursos, ya que los proyectos serán cada vez más complejos y a su vez más funcionales. Una de las formas que se tiene para simplificar esto es el uso de librerías, que, en el fondo, son una colección de funciones que nos permiten obtener resultados o ejecutar acciones a partir de instrucciones para una funcionalidad especifica. Por ejemplo, cuanto nos facilita la vida el uso de la librería para mover un servomotor, o cuando realizamos la medición de temperatura y humedad con un DHT, las librerías incluidas en el sketch permiten a los elementos la capacidad de ser fácilmente usados, algunas librerías han sido incluidas y ya no necesitan ser declaradas, por ejemplo, hace unos años era necesario declarar la librería “math.h” para usar expresiones trigonométricas, hoy las podemos usar sin declarar la librería.

Recordemos un poco, al abrir el IDE de Arduino para escribir nuestro programa en el Sketch encontraremos 3 bloques con funciones específicas. De arriba abajo encontraremos un bloque de declaración, otro con el void setup y como último el void loop:

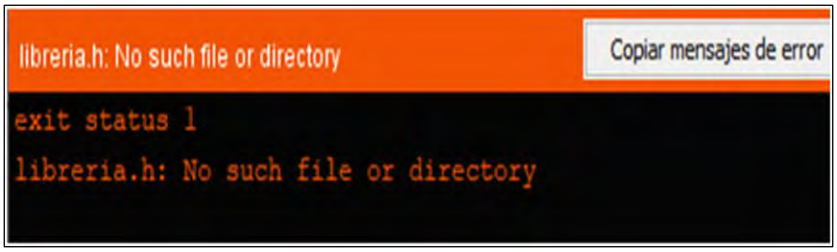
Figura 31. Cuadro de dialogo muestra el programa escrito en el sketch con sus declaraciones específicas.



Fuente: Sistema Arduino

El bloque de declaración, que es todo lo anterior al setup, es un espacio que utilizaremos para, principalmente, definir las librerías que se usara en el programa. Las librerías se definen de la forma #include <libreria.h>. si la librería no se encuentra en nuestro repositorio, esto significa que no existía en las librerías por defecto en el IDE, no las descargamos desde la gestión de librerías ni las agregamos de forma manual, a la hora de compilar nuestro programa tendremos un error donde nos indicara que la librería no es un archivo o un directorio.

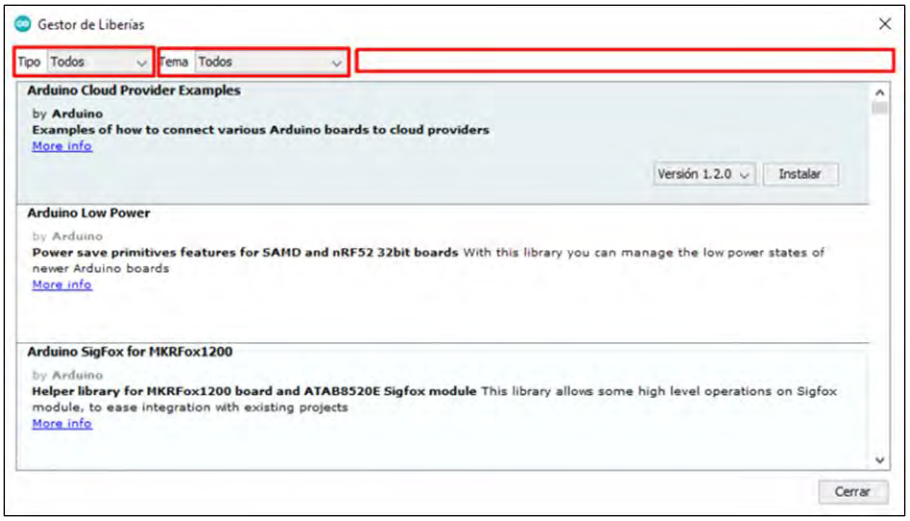
Figura 32. Ejemplo cuando no hay una librería definida.



Fuente: Sistema Arduino

A grandes rasgos el bloque void setup permite inicializar parámetros y se ejecuta una sola vez y el bloque void loop es el que repetirá interminablemente el programa. Para evitar ese error y poder utilizar las librerías de la mejor manera, se descarga directamente desde el repositorio oficial de Arduino a través del IDE. Para ello abrimos el IDE de Arduino, seleccionamos el menú "Herramientas" y dar click en "Administrar Bibliotecas…" o se presiona las teclas Ctrl+Shift+i. otra forma de hacerlo es ingresar al menú programa, sub menú Incluir Librería, y dar click en "Administrar Bibliotecas…". Con esto se abrirá la pestaña del Gestor de Librerías, allí podemos navegar y encontrar muchas librerías, filtrar por tipo, tema o buscar directamente la que necesitemos.

Figura 33. Gestor de librerías



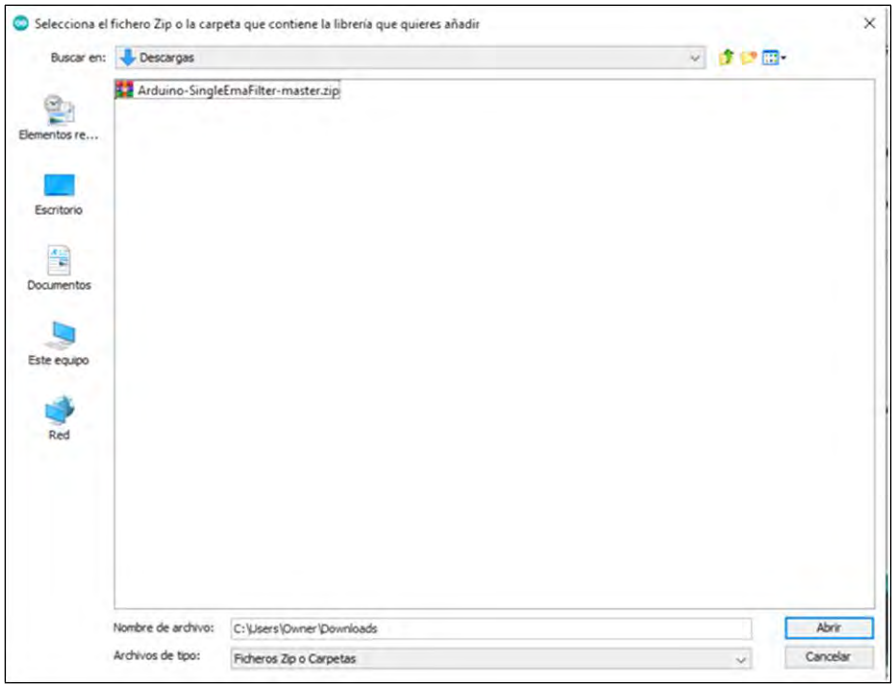
Fuente: Sistema Arduino

Una vez identificada la librería, se selecciona la versión y presionamos instalar. Con esto la librería será incorporada a nuestro repositorio de librerías y ya se puede usar, para ello vamos al menú Programa, Incluir librería y seleccionamos la biblioteca a usar. Por lo general se coloca al final, en la sección "Contribución Bibliotecas" o "Recomendado Bibliotecas".

En ocasiones, dependiendo del desarrollo, puede que la librería no la encontremos entre las que nos entrega el gestor de librerías, por ello se dejó un espacio para poder incluir librerías externas que pueden haber sido desarrollas por terceros o por nosotros.

El procedimiento es similar al anterior. En este caso ingresamos a programa, Incluir librería y damos click en la segunda opción, Añadir Biblioteca Zip. Se desplegará una ventana como la siguiente:

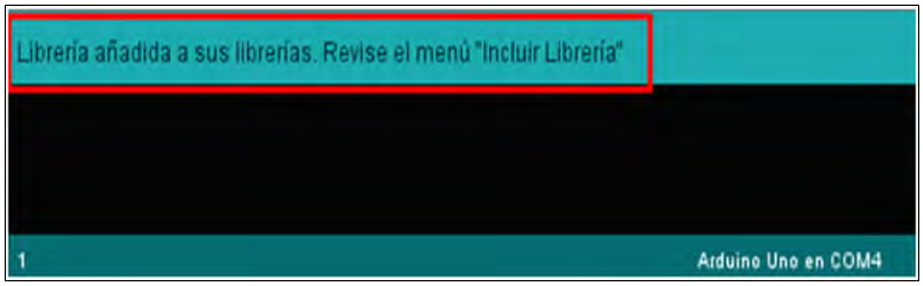
Figura 34. Ventana muestra la selección de librerías



Fuente: Sistema Arduino

Buscamos la librería descargada, que debe estar en formato de compresión Zip y damos click en abrir. Si la librería tiene el formato y los archivos correctos tendremos un mensaje de esta forma.

Figura 35. Dialogo que muestra la librería incluida



Fuente: Sistema Arduino

En ocasiones al incorporar librerías mediante archivo. Zip se encontrarán con fallos y las librerías no podrán ser incorporadas. Esto se indicará con un mensaje naranja.

Los archivos que una librería debe contener en el Zip son específicos y Arduino establece que deben estar en este formato de compresión para ser capaz de importarlo. Lo que encontramos en el interior del zip es:

- Fichero .h, por headers, para constantes, includes y definiciones de clase
- Fichero .cpp, suele estar el peso del programa, funciones de clase, propiedades y funciones de apoyo
- Fichero .c, funciones y programas en C. Podría no estar incluido
- Fichero keywords.txt, define la librería
- Fichero README, incluye todo el detalle de la librería
- Directorio de ejemplos, examples.

Cabe mencionar que cada librería nos entregara la posibilidad de utilizar comandos específicos, y únicos, por ello siempre deben intentar conseguir la información del autor para conocer la sintaxis y la aplicación específica de cada uno de ellos. Por ejemplo, la librería para el servomotor incluye un programa que permite mover el servo con un potenciómetro en ángulos de 0 a 180 grados. (Electronics, s.f.)

CAPÍTULO 2.

SENSORES BÁSICOS COMPATIBLES CON ARDUINO UNO.

Un sensor es un objeto cuyo propósito es detectar eventos o cambios en su entorno, y luego proporcionar una salida correspondiente.

Un sensor proporciona datos a un sistema a través de su entrada. Por ejemplo, se puede ajustar un sistema de aire acondicionado para que la temperatura ambiente sea de 15 grados, teniendo en cuenta que anteriormente esta desactivada (y que la temperatura actual de la habitación es diferente de los 15 grados), ahora el sistema comienza a bombear aire para que la habitación alcance la temperatura deseada, ¿cómo sabe cuándo se ha alcanzado la temperatura ambiente? Aquí es donde entran en juego el sensor de Arduino, debe haber un sensor en la habitación para indicar al sistema de aire acondicionado que detenga el bombeo cuando se alcance la temperatura deseada, en este ejemplo los 15grados.

Las señales analógicas nos permiten medir con precisión las temperaturas, las distancias, a través de un rango y luego responder en consecuencia. Mientras que las señales digitales sólo nos dicen si una señal está activa o no.

Los sensores para Arduino se identifican muy fácilmente con los voltajes lógicos. Arduino utiliza voltajes lógicos TTL de 5VDC. Los microcontroladores utilizados en el Arduino Uno se basan en la arquitectura AVR. Entonces si un sensor es compatible, este tendrá de salida los voltajes que acepta Arduino.

Para conocer que voltajes lógicos son aceptados por los sensores para Arduino Uno tenemos que buscar en la hoja de datos de los microcontroladores ATmega328 buscar la tabla de características eléctricas.

Los sensores para Arduino, serán compatibles, si estos pueden funcionar con sus voltajes lógicos. Por ejemplo, si Arduino escribe un "1" digitalmente, esto es: digitalWrite (PIN, HIGH), el sensor que sea compatible deberá poder reconocer el voltaje que proporciona la tarjeta de Arduino.

En el caso del Arduino uno, los sensores deberán admitir mínimo un voltaje de 4.1 Vdc y máximo 5Vdc. Y en el caso contrario, si el sensor proporciona un voltaje, este voltaje deberá estar entre lo que puede aceptar el microcontrolador. A este dato se le conoce como parámetro VIH (Voltage Input High) en español es el Alto Voltaje de Entrada. En la tabla antes mencionada este parámetro equivale a: 0.6 Vcc como mínimo y Vcc+0.5 Vdc como máximo, esto es, 3 Vdc mínimo (Vcc=5 Vdc) Y 5.5 Vdc como máximo.

En resumen para que un sensor sea compatible para Arduino, tiene que generar voltajes lógicos de entre 3 Vdc y 5.5 Vdc de salida y poder admitir al menos 4.1V como entrada. Esto es para los valores lógicos altos. Para los valores lógicos en bajo, -0.5 Vdc a 0.1Vcc y aceptar valores lógicos bajos de hasta 1 Vdc.

Proceso para verificar compatibilidad de sensores para Arduino.

- Verificar que el voltaje de alimentación del sensor Vcc sea de 5Vdc y/o 3.3Vdc.
- Verificar que el sensor tenga voltaje lógico con 5Vdc.
- Usar el código o la hoja de datos del microcontrolador para el sensor con Arduino.
- Utilizar los circuitos del tipo de sensores en conjunto con Arduino.

Nomenclatura:

- Vcc: alimentación del Arduino.
- Vdc: Voltaje de corriente directa.
- VIH: Voltaje de entrada en alto (0.6 Vcc a Vcc+ 0.5 V dv)
- VIL: Voltaje de entrada en bajo (-0.5 Vdc a 0.1 Vcc).
- VOL: Voltaje de salida en bajo (máximo 1.0 Vdc)
- VOH: Voltaje de salida en alto (mínimo 4.1 Vdc)

(Dieguez, 2020)

I. Sensor ultrasónico con Arduino

La necesidad de censar nuestro entorno y saber si en frente hay un obstáculo y a que distancia se encuentra, el sensor HC-SR04 nos permite hacer eso. A continuación vamos a conocer de forma detallada las características del sensor HC-SR04, como calibrar el sensor, como conectarlo y como utilizarlo con Arduino.

El sensor HC-SR04 es un sensor de distancia de bajo costo, su uso es muy frecuente en la robótica, utiliza transductores de ultrasonido para detectar objetos.

El sensor ultrasónico HC-SR04 es un módulo electrónico que incorpora un par de transductores de ultrasonido que se utilizan de manera conjunta para determinar la distancia entre el sensor (módulo) y un objeto. Uno de los transductores actúa como emisor de ultrasonido, mientras que el otro recibe el eco o rebote de la señal original.

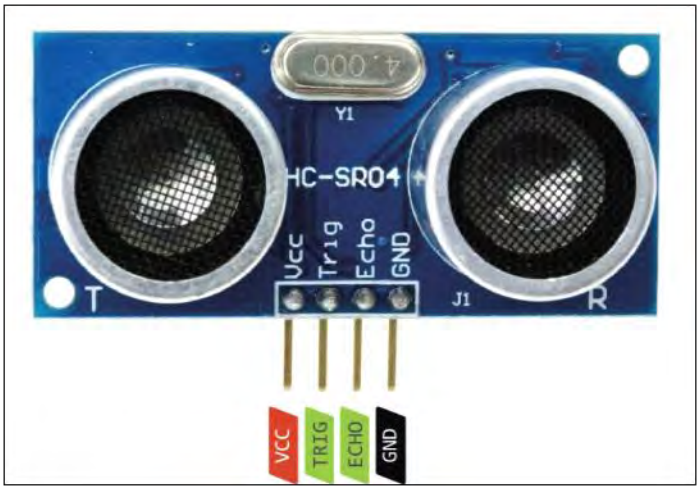
Su funcionamiento consiste al recibir el pulso de disparo, un transductor emite una "ráfaga" de ultrasonido y el otro capta el rebote de dicha onda sobre el objeto detectar. El tiempo que tarda la onda sonora en ir y regresar a un objeto puede utilizarse para conocer la distancia que existe entre el origen del sonido y el objeto.

El sensor ultrasónico tiene dos versiones: el PING (3-pin) y el HC-SR04 (4-pin). Ambas versiones tienen pines GND y Vcc. El HC-SR04 tiene dos pines de señal separados: uno para el transmisor (Trigger) y otro para el receptor (Eco), Estos pines de señal están conectados a los pines digitales del Arduino Uno.

El sensor ultrasónico HC-SR04 funciona con 4 pines: GND, Vcc, Trigger (OUTPUT), y Eco (Echo). Como se ilustra en la imagen 2.1. Este posee los pines por separado, lo único en lo que se debe preocupar es el tiempo que se transmitirá y el emisor cuento tiempo debe esperar para conocer la distancia.

Versión de sensor ultrasónico de distancia HC-SR04 compatibles con Arduino.

Figura 36. Módulo Sensor ultrasónico HS-RS04



Fuente: Sistema Arduino

Tabla 5. Tabla de características del sensor Ultrasónico HC-SR04.

sensor ultrasónico HC-SR04

Modelo:	HCSR04	Voltaje de alimentación:	5 VDC	Angulo de medición:	15º	Rango de tiempo de señal de eco:	100 uS a 25000 uS
Marca:	Genérico (sensores)	Corriente típica en operación:	15 mA	Conector:	Header macho estándar 0.1 pulgadas	Tiempo entre medidas:	20 mS
Tipo de sensor:	proximidad, distancia, presencia	Rango de medición	: 2 cm a 400 cm	Interfaz de 4 conexiones	Vcc, Trigger, Echo y Gnd.	Peso:	9 gramos
Principio de funcionamiento:	Ultrasonido	Frecuencia del pulso ultrasónico:	40 KHz	Tiempo en alto para señal de disparo:	10uS	Dimensiones:	45mm x x 20mm x 15mm.

Fuente: Elaboración propia

La conexión del sensor HC-SR04 con Arduino es muy sencilla y la podemos realizar utilizando un protoboard, o directamente con alambres. Para lograr que el sensor funcione correctamente, son necesarias estas 4 señales:

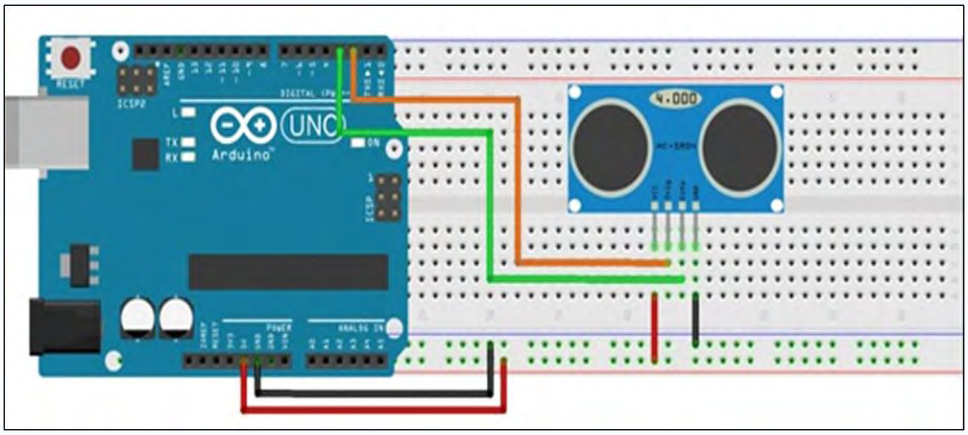
- Alimentación de 5 volts.
- Tierra o común del circuito.
- Señal de disparo (trigger).
- Señal de eco (echo).

Empezamos insertando el sensor ultrasónico en un protoboard y con cables hacemos las siguientes conexiones:

- Trigger del sensor al pin 2 del Arduino uno.
- Echo del sensor al pin 3 de Arduino uno.

Diagrama de pines para Arduino Uno que tiene la misma función al igual que la versión PING.

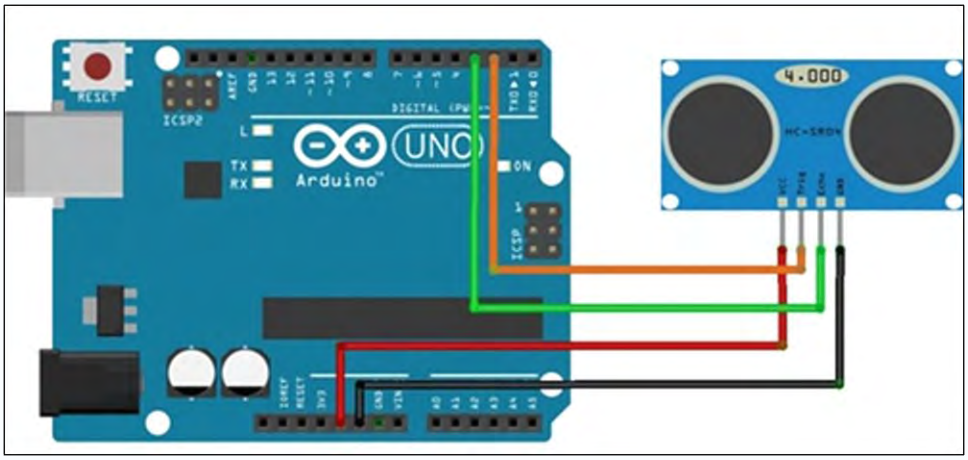
Figura 37. Diagrama de conexión del sensor HC-SR04 con Arduino uno



Fuente: Sistema Arduino

También puede conectarse el sensor al módulo Arduino directamente:

Figura 38. Diagrama de conexión del sensor HC-SR04 con Arduino uno



Fuente: Sistema Arduino

Todas las conexiones se realizan con el Arduino Apagado (desconectado de la PC o de cualquier fuente externa):

Ahora abrimos el entorno de programación de Arduino, en Herramientas ->Tarjeta, y seleccionamos el modelo de placa Arduino Uno que estemos utilizando. (Mechatronic, s.f.)

Código para el funcionamiento de sensor HC-SR04.

Presentamos el código para probar su funcionamiento, los pines de conexión deben coincidir perfectamente para que el sensor HC-SR04 y Arduino puedan funcionar en conjunto, de lo contrario, podemos causar daños y sobrecalentamiento a cualquiera de las dos placas: Arduino o sensor.

Una vez configurado el IDE, empezamos a programar nuestro sketch, configuramos los pines y la comunicación serial a 9800 baudios

```
const int Trigger = 2;   //Pin digital 2 para el Trigger del sensor
const int Echo = 3;   //Pin digital 3 para el echo del sensor

void setup() {
  Serial.begin(9600);//inicializamos la comunicación
  pinMode(Trigger, OUTPUT); //pin como salida
  pinMode(Echo, INPUT);  //pin como entrada
  digitalWrite(Trigger, LOW);//Inicializamos el pin con 0
}
```

Ahora en el bucle void loop() empezamos enviando un pulso de 10us al Trigger del sensor

```
  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);          //Enviamos un pulso de 10us
digitalWrite(Trigger, LOW);
```

Seguidamente recibimos el pulso de respuesta del sensor por el pin Echo, para medir el pulso usamos la función pulseIn(*pin*, *value*)

```
  t = pulseIn(Echo, HIGH); //obtenemos el ancho del pulso
```

$$Velocidad = \frac{distancia\ recorrida}{tiempo}$$

Donde Velocidad es la velocidad del sonido 340m/s, pero usaremos las unidades en cm/us pues trabajaremos en centímetros y microsegundos, **tiempo** es el tiempo que demora en llegar el ultrasonido al objeto y regresar al sensor, y la ***distancia recorrida*** es dos veces la distancia hacia el objeto

Finalmente enviamos serialmente el valor de la distancia y terminamos poniendo una pausa de 100ms, que es superior a los 60ms recomendado por los datos técnicos del sensor

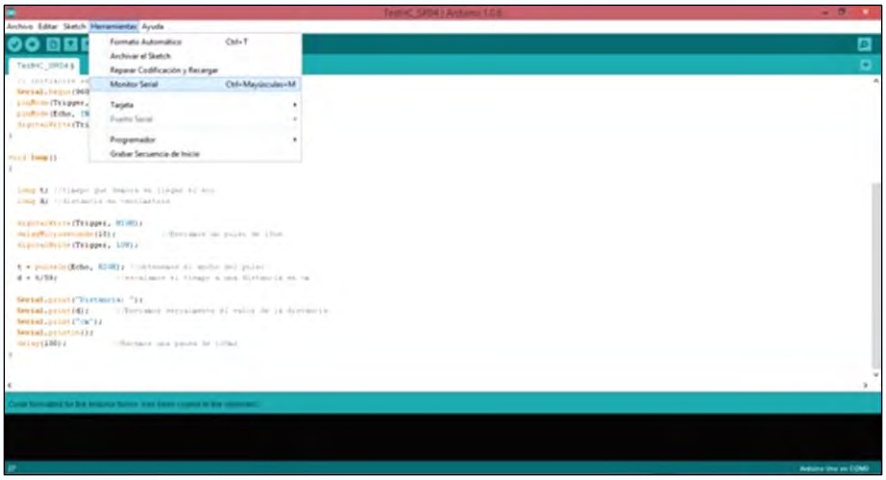
A continuación, se muestra el código completo del programa.

```
const int Trigger = 2;   //Pin digital 2 para el Trigger del sensor
const int Echo = 3;   //Pin digital 3 para el Echo del sensor

void setup() {
  Serial.begin(9600);//iniciailzamos la comunicación
  pinMode(Trigger, OUTPUT); //pin como salida
  pinMode(Echo, INPUT);  //pin como entrada
  digitalWrite(Trigger, LOW);//Inicializamos el pin con 0
}
void loop()
{
  long t; //timepo que demora en llegar el eco
  long d; //distancia en centimetros

  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);          //Enviamos un pulso de 10us
  digitalWrite(Trigger, LOW);

  t = pulseIn(Echo, HIGH); //obtenemos el ancho del pulso
  d = t/59;             //escalamos el tiempo a una distancia en cm

  Serial.print("Distancia: ");
  Serial.print(d);      //Enviamos serialmente el valor de la distancia
  Serial.print("cm");
  Serial.println();
  delay(100);          //Hacemos una pausa de 100ms
}
```

Conecte el Arduino Uno y cargue el programa.

Después de esto el Arduino y sensor ya deben estar trabajando, para poder visualizar los datos vaya a herramientas y habrá el monitor serial. Como se muestra la imagen 2.5 a continuación.

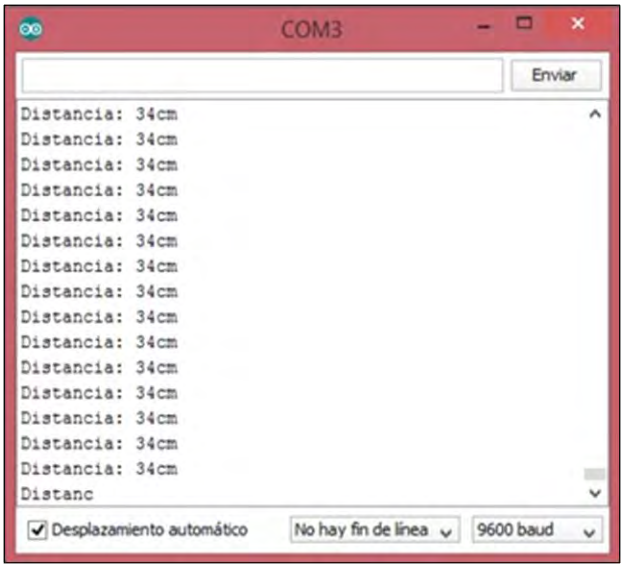
Figura 39. Cuadro de dialogo IDE selecciona monitor serial



Fuente: Sistema Arduino

En el monitor serial le aparecerán los valores de la distancia que censa el HC-SR04, ponga un objeto al frente y varíe su distancia respecto al sensor y verifique que la distancia mostrada en el monitor serial sea la correcta. (Mechatronic, s.f.)

Figura 40. IDE muestra la distancia obtenida del sensor ultrasónico HC-SR04



Fuente: Sistema Arduino

.

II. Sensor ultrasónico versión PING (3-pin)

Versión PING (3 pin) transductor de medición compatible con Arduino Uno.

Figura 41. Sensor ultrasónico versión PING (3 pin)

Fuente: Sistema Arduino

El sensor ultrasónico PING funciona con 3 pines: GND, Vcc y Señal. Como se muestra en la imagen 2.7.

El sensor de ultrasónico PING es un magnífico sensor capaz de alcanzar distancias entre 0 a 5 metros, el sensor envía un sonido fino y pequeño que el oído humano no puede recibir, al hacer un choque con un objeto, se regresa al sensor, a este fenómeno se le llama eco, cuando el emisor del sensor capta el sonido, solo mide la duración del 0 y tiempo que tardo el 1 (que es el sonido), al saber la cantidad del tiempo y el número de ceros, el microcontrolador sabrá la distancia recorrida del sonido y el tiempo que tardo en regresar, a este paso se llama "Give up" que en español es rendición.

La versión PING tiene una clavija de señal dual que puede ser usada tanto como un INPUT como un OUTPUT, ya que el mismo pin de señal funciona para entrada y salida, tan solo se debe de tomar en cuenta los microsegundos que estará enviando y recibiendo la señal, ya que puede arruinar tanto el sensor como el Arduino.

Tabla 6. características del sensor Ultrasónico PING

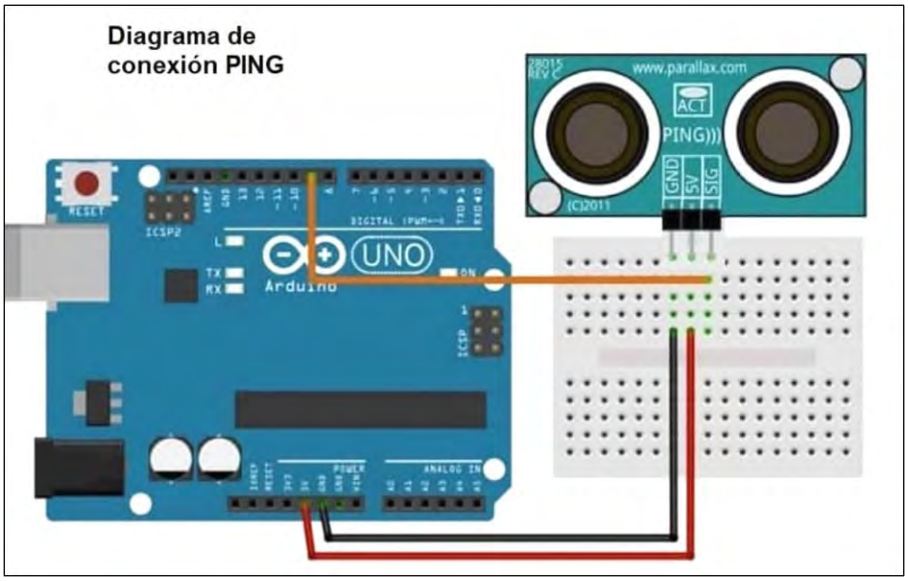
Sensor ultrasónico modulo PING	
Precisión de distancia de 2cm a 3m	*no requiere de soldadura*
Comunicación simple con pulso de entrada/ pulso de salida	*Requisitos de alimentación: 5 VDC*
El indicador LED muestra la medición en curso	*Comunicación: Positivo TTL pulso*
Consumo de energía de 20 mA	*Dimensiones: 0,81 x 1,8 x 0,6 en (22 x 46 x 16 mm)*
Angulo de aceptación estrecho	*Rango de temperatura: 32 a 158 ° F (0 a 70 ° C)*
cabezal de 3 pines facilita la conexión mediante un cable de extensión	*Hoja de Datos: 28015-PING-v1.6.pdf*

Fuente: Elaboración Propia

La conexión del sensor PING al sistema Arduino es sencillo, el sensor cuenta con tres pines: GNP, 5V y la señal (SIG). El siguiente circuito le permite conectar rápidamente el sensor PING utilizando un protoboard. El pin GND del módulo PING se conecta al GND del Arduino, el pin 5V al Vdd, y el pin SIG se conecta al pin I/O P9. La imagen 2.8 ilustra dicha conexión. Este circuito funciona con Basic Stamp, El programa se muestra más adelante.

El sensor PING lo puede conectar de acuerdo al siguiente diagrama:

Figura 42. Diagrama de conexión del sensor ultrasónico versión PING (3 pin)



Fuente: Sistema Arduino

Este programa se realiza en BASIC Stamp 2 muestra medidas de distancia en pulgadas y centímetros en la Terminal de Depuración del BASIC Stamp.

```
Mide la distancia con el sensor Ping y la pantalla en pulgadas y en cm.
' {$STAMP BS2}
' {$PBASIC 2.5}
Constantes de conversión para mediciones de temperatura ambiente.
CmConstant CON 2260
InConstant CON 890
cmDistance VAR Word
inDistance VAR Word
time VAR Word
DO
 PULSOUT 9, 5
 PULSIN 9, 1, time
 cmDistance = cmConstant ** time
 inDistance = inConstant ** time
 DEBUG HOME, DEC3 cmDistance, " cm"
 DEBUG CR, DEC3 inDistance, " in"
 PAUSE 100
LOOP
```

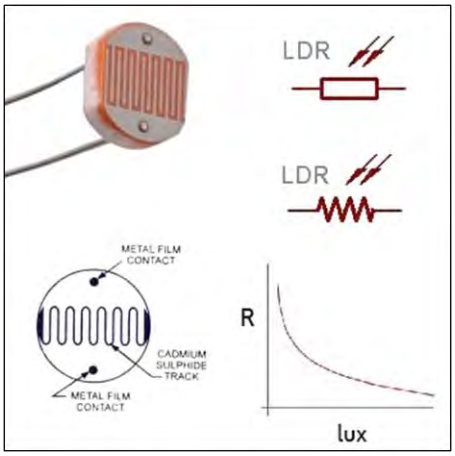
Código en Arduino IDE para sensor ultrasónico versión PING (3 pin)

```
unsigned long pulsos; // variable para medir el pulso.
float distancia;//calcularemos la distancia.
int salida = 9; // Donde enviará el sonido.

void setup(){Serial.begin(9600);}
void loop(){
  pinMode(salida,OUTPUT);//declaramos el 9 como salida.
  digitalWrite(salida,HIGH);//escribiremos un 1.
  delayMicroseconds(10);//esperaremos los microsegundos.
  digitalWrite(salida,LOW); // escribiremos un 0.
  pinMode(salida, INPUT);//lo declaramos como salida.
    /*se mide el ancho de pulso y de espera transmitida
  del eco del sonido.*/
  pulsos = pulseIn(salida, HIGH);
  Serial.print("Tiempo = ");
  /*Lo transformamos el pulso en tiempo recibido*/
  Serial.print(float(pulsos/1000.0));
  Serial.print("ms, distancia = ");
  //transformamos el pulso en distancia.
  distancia = ((float(pulsos/1000.0))*34.32)/2;
  Serial.print(distancia);
  Serial.println("cm");
  delay(100);
}
```

https://artbanshee.jimdofree.com/arduino/sensor-ultras%C3%B3nico/

III. Sensor Fotoresistor con Arduino.

Figura 43. Imagen fotorresistencia o LCD



Fuente: Sistema Arduino

Un fotoresistor, o LDR (light-dependent resistor) es un dispositivo cuya resistencia varía en función de la luz recibida. Podemos usar esta variación para medir, a través de las entradas analógicas, una estimación del nivel de luz. Este sensor de Fotoresistor es básico.

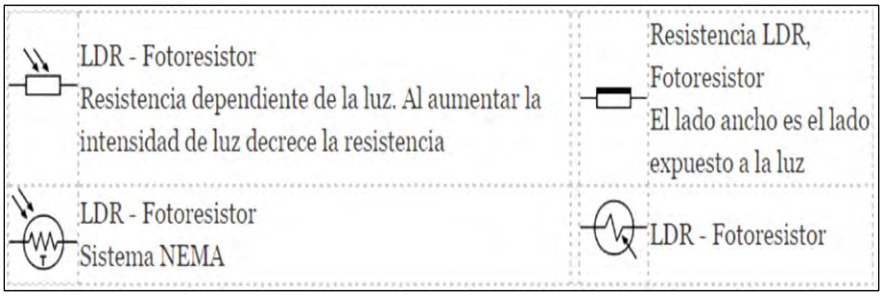
Un fotoresistor está formado por un semiconductor, típicamente sulfuro de cadmio CdS. Al incidir la luz sobre él algunos de los fotones son absorbidos, provocando que electrones pasen a la banda de conducción y, por tanto, disminuyendo la resistencia del componente.

Por tanto, un fotoresistor disminuye su resistencia a medida que aumenta la luz sobre él. Los valores típicos son de 1 MΩ en total oscuridad, a 50-100 Ohm bajo luz brillante.

El cuarto sensor analógico de Arduino es una fotorresistencia (también conocida como resistencia dependiente de la luz o LDR). Los fotoresistores pueden detectar el brillo. Si quieres construir un robot que siga una luz o un prototipo que dispare una acción basada en condiciones de luz u oscuridad, deberás usar un fotoresistor.

Los fotoresistores tienen dos pines (Tierra y señal). Debido a que un fotoresistor es un tipo especial de resistencia, crearás un divisor de voltaje contra la señal. También usarás un resistor 10KΩ para cambiar la señal hacia arriba o hacia abajo. Esto determinará si la oscuridad es una lectura alta o baja.

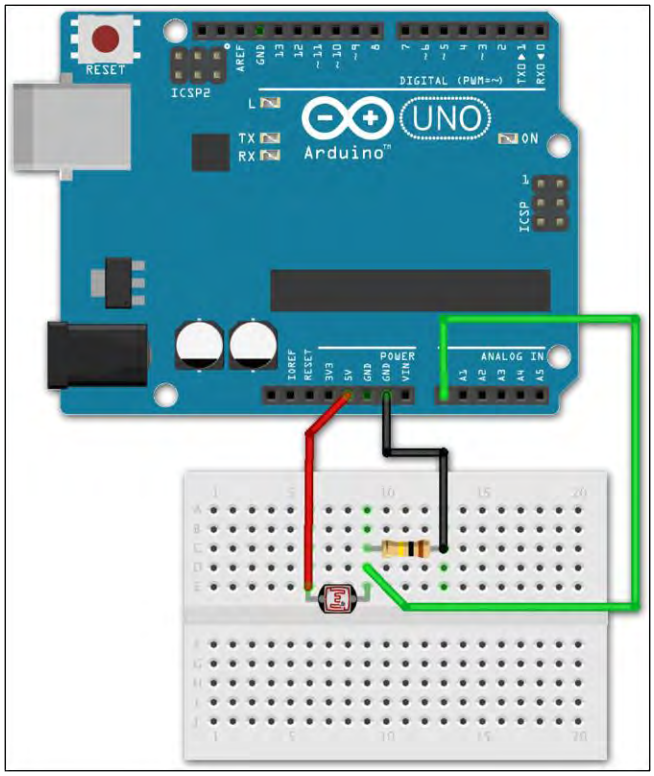
Figura 44. Características de fotoresistores.



Fuente: Sistema Arduino

Conecta el resistor a tierra si deseas que los valores bajos representen la oscuridad y los valores altos representen el brillo. Conecta la resistencia a la corriente si deseas cambiar esta métrica y hacer que los valores de luz representen una lectura baja. (Llamas, 2015)

Figura 45. Conexión del sensor analógico básico fotoresistor con Arduino uno



Fuente: Sistema Arduino

Conexión del dispositivo

La conexión de cualquier dispositivo a la tarjeta Arduino se realiza conectándolo a los contactos situados en la tarjeta controladora: uno de los pines digitales o analógicos o los pines de alimentación. Un simple LED puede conectarse mediante dos contactos: tierra (GND) y señal (o contacto de potencia).

El sensor más simple requiere un mínimo de tres contactos: dos para la fuente de alimentación y uno para la señal, para la conexión a un dispositivo externo, recuerde que la tarjeta sólo puede utilizarse como fuente de alimentación si el dispositivo no consume más del límite de corriente permitido por el controlador.

Ejemplos de código

Presentamos dos ejemplos de código. A continuación utilizamos las entradas digitales para hacer parpadear el LED integrado en la placa mientras el LDR recibe luz suficiente.

```
const int LEDPin = 13;
const int LDRPin = 2;
void setup()
{
pinMode(LEDPin, OUTPUT);
pinMode(LDRPin, INPUT);
}
void loop()
{
  int value = digitalRead(LDRPin);
  if (value == HIGH)
  {
    digitalWrite(LEDPin, HIGH);
    delay(50);
    digitalWrite(LEDPin, LOW);
    delay(50);
  }
}
```

El siguiente código proporciona una lectura del nivel de iluminación recibido. Observar que los cálculos se realizan con aritmética de enteros, evitando emplear números de coma flotante, dado que ralentizan mucho la ejecución del código. (Llamas, 2015)

```
const long A = 1000;     //Resistencia en oscuridad en KΩ
const int B = 15;        //Resistencia a la luz (10 Lux) en KΩ
const int Rc = 10;       //Resistencia calibración en KΩ
const int LDRPin = A0;   //Pin del LDR
int V;
int ilum;
void setup()
{
  Serial.begin(115200);
}
void loop()
{
  V = analogRead(LDRPin);

  //ilum = ((long)(1024-V)*A*10)/((long)B*Rc*V);  //usar si LDR entre GND y A0
 ilum = ((long)V*A*10)/((long)B*Rc*(1024-V));     //usar si LDR entre A0 y Vcc (como en el esquema anterior)

  Serial.println(ilum);
  delay(1000);
}
```

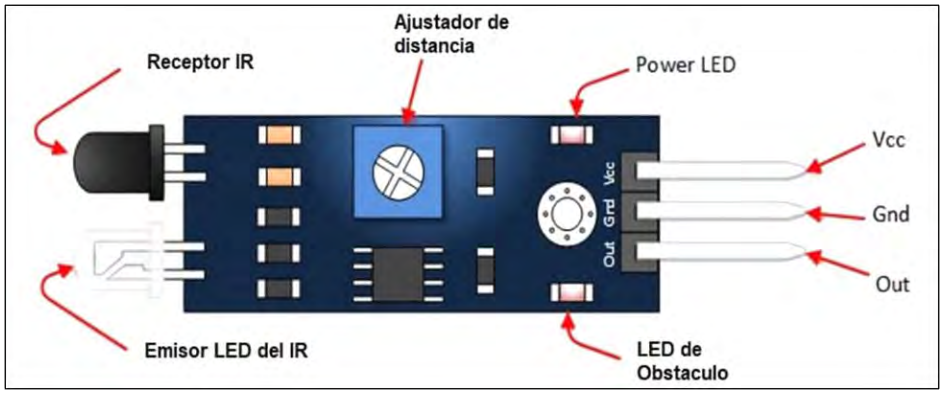
IV. Sensor de emisión de infrarrojos

Sensor de emisión infrarrojo, también llamado «diodo emisor de infrarrojos». Funciona a una señal moduladora de 38KHz. Este sensor puede ser usado para enviar código a otro Arduino o para controlar un televisor.

Un detector de obstáculos infrarrojo es un dispositivo que detecta la presencia de un objeto mediante la reflexión que produce en la luz. El uso de luz infrarroja (IR) es simplemente para que esta no sea visible para los humano, son sensores sencillos. Se dispone de un LED

emisor de luz infrarroja y de un fotodiodo (tipo BPV10NF o similar) que recibe la luz reflejada por un posible obstáculo.

Figura 46. Sensor de emisión infrarrojo



Fuente: Sistema Arduino

La mayoría de los sensores IR para evitar obstáculos son muy económicos, lo que los convierte en una opción accesible para proyectos sencillos. Aquí tienes en primer plano un módulo de sensor de infrarrojos en la Imagen 2.10.

Características:

Tabla 7. Módulo de sensor IR infrarojo

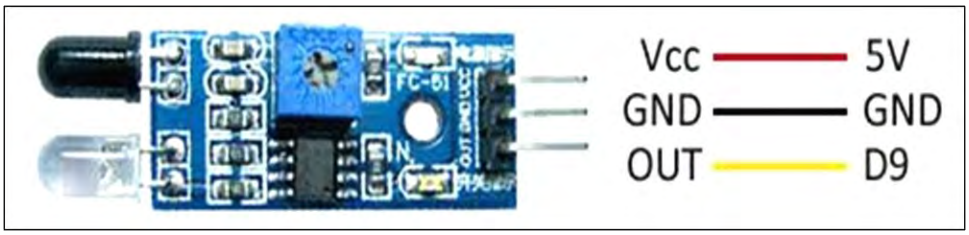
Módulo de sensor IR Infrarrojo.	
Marca: Ismart	*Rango de operación: 2 a 30cm, con un ángulo de 35 grados*
Sensor IR Switch Modulo.	*Voltaje : 3-5V DC*
Peso del paquete: 0.1 kg	

Fuente: Elaboración propia

Este tipo de sensores actúan a distancias cortas, típicamente de 5 a 20mm. Además la cantidad de luz infrarroja recibida depende del color, material, forma y posición del obstáculo, por lo que no disponen de una precisión suficiente para proporcionar una estimación de la distancia al obstáculo.

Pese a esta limitación son ampliamente utilizados para la detección de obstáculos en pequeños vehículos o robots. Su bajo coste hace que sea frecuente ubicarlos en el perímetro, de forma que detectemos obstáculos en varias direcciones. También son útiles en otro tipo de aplicaciones como, por ejemplo, detectar la presencia de un objeto en una determinada zona, determinar una puerta está abierta o cerrada, o si una máquina ha alcanzado un cierto punto en su desplazamiento. (Prometec)

Figura 47. Diagrama de conexión del Sensor emisor IR infrarrojo con Arduino



Fuente: Sistema Arduino

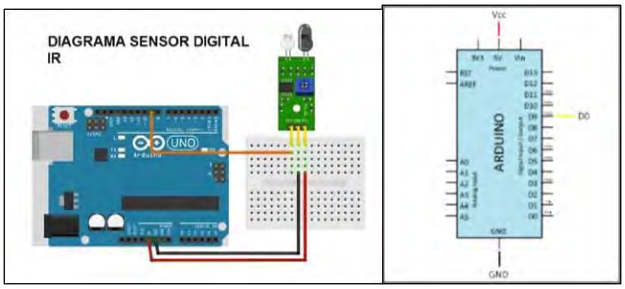
Esquema eléctrico

El montaje es sencillo, la alimentación del módulo es a través de Vcc y GND conectándolos, respectivamente, a la salida de 5V y GND en Arduino.

Finalmente, conectamos la salida digital del sensor a una entrada digital para leer el estado del sensor.

Finalmente, conectamos la salida digital del sensor a una entrada digital para leer el estado del sensor. como se muestra en la imagen 2.12.

Figura 48. Imagen de conexión del sensor infrarrojo con el Arduino uno



Fuente: Sistema Arduino

Opcionalmente, se calibra el umbral de disparo acercando un objeto al detector de obstáculos y se regula la salida digital con el potenciómetro. Si se desea omitir este paso, dejar el potenciómetro en un valor medio.

El código es igualmente sencillo. Simplemente leemos el estado de la entrada digital, tal y como vimos en la entrada Entradas digitales en Arduino.

Si el sensor se dispara, ejecutamos las acciones necesarias. (Prometec)

Código de programación.

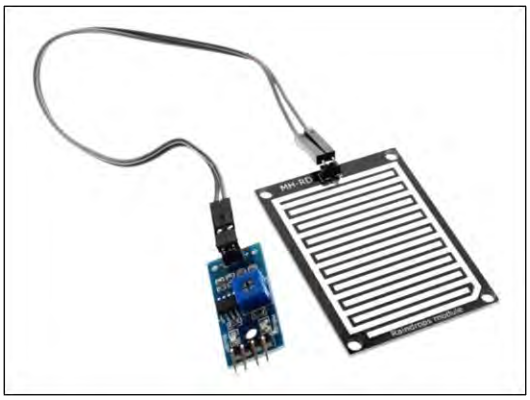
```
const int sensorPin = 9;

void setup() {
  Serial.begin(9600);   //iniciar puerto serie
  pinMode(sensorPin , INPUT);  //definir pin como entrada
}

void loop(){
  int value = 0;
  value = digitalRead(sensorPin );  //lectura digital de pin

  if (value == HIGH) {
     Serial.println("Detectado obstaculo");
  }
  delay(1000);
}
```

https://www.luisllamas.es/detectar-obstaculos-con-sensor-infrarrojo-y-arduino/

V. Sensor de lluvia.

Este sensor detecta la presencia de lluvia por la variación de conductividad del sensor al entrar en contacto con el agua, Constructivamente son sensores sencillos. Se dispone de dos contactos unidos a unas pistas conductoras entrelazadas entre sí a una pequeña distancia, sin existir contacto entre ambas. Al depositarse agua sobre la superficie, se pone en contacto eléctrico ambos conductores, lo que puede ser detectado por un sensor.

Figura 49. Sensor de lluvia HL-83.



Fuente: Sistema Arduino

Los valores analógicos medidos varían desde 0 para una placa totalmente empapada, a 1023 para una placa totalmente seca. La salida digital dispara cuando el valor de humedad supera un cierto valor, que se ajusta mediante el potenciómetro. Por tanto, se obtendrá una señal LOW en ausencia de lluvia, y HIGH con presencia de lluvia.

El sensor de lluvia puede ser empleado, por ejemplo, para extender un toldo o activar algún otro mecanismo, hacer sonar una alarma, o registrar la cantidad de tiempo (días, horas) en el que se producen precipitaciones en una determinada zona. No podemos medir la cantidad de lluvia, solo su presencia.

El sensor de lluvia también puede ser empleado para detectar inundaciones, colocándolo en el suelo de un sótano o sala de calderas, puede ser útil para detectar cuando el agua de un depósito sobrepasa un determinado nivel. (Llamas, DETECTOR DE LLUVIA CON ARDUINO Y SENSOR FC-37 O YL-83, 2018)

Características:

Tabla 8. Sensor de lluvia HL-83

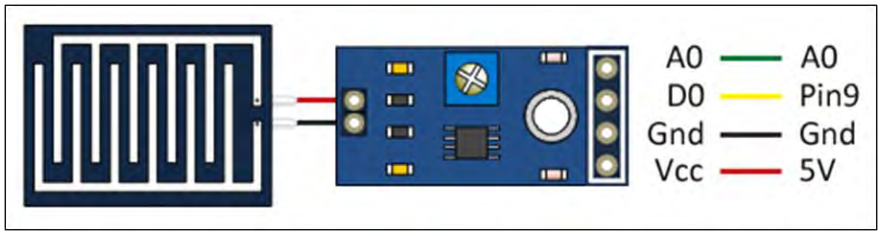
Sensor de lluvia HL-83	
Marca: ELECAN 3D	Sensibilidad ajustable por potenciómetro
Modelo: HL-83.	Modo de salida dual: analógica y digital
Microcontrolador: LM-393	LED rojo – indicador de encendido
Voltaje máximo de entrada recomendado: 5V.	LED verde – indicador de salida de conmutación digital
Voltaje mínimo de entrada recomendado: 3.3V	Medidas de sensor: 5.4 x 3.9
Voltaje de funcionamiento: 3.3V- 5V.	Medidas PCB: 3 x 1.5 cm

Fuente: Elaboración propia

Este sensor cuenta con doble material de alta calidad FR-04, tratamiento de niquelado en su superficie, protección contra oxidación, tiene un periodo de vida con un rendimiento superior, en contraste a modelos similares.

Diagrama eléctrico el sensor de lluvia HL-83.

Figura 50. diagrama eléctrico del sensor de lluvia HL-83.

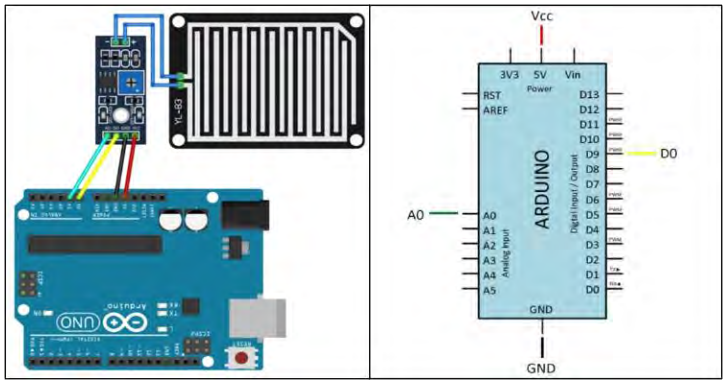


Fuente: Sistema Arduino

Esquema de montaje

El esquema eléctrico es sencillo. Conectamos el sensor a la placa de medición. El sensor no tiene polaridad, por lo que puede ser conectada en cualquier sentido.

Figura 51. Esquema de conexión del sensor de lluvia HL-83 con Arduino uno.



Fuente: Sistema Arduino

Por otro lado, alimentamos la placa de medición conectando los pines GND y 5V a los pines correspondientes de Arduino.

Si quisiéramos obtener el valor analógico de la medición, conectaríamos la salida analógica del sensor a una entrada analógica de Arduino, aunque como hemos dicho este sensor no dispone de la precisión suficiente para que el valor analógico sea realmente de utilidad.

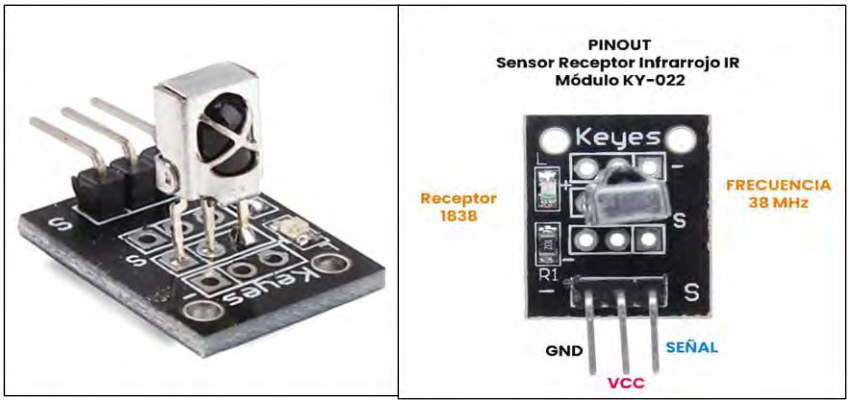
Opcionalmente, podemos calibrar el umbral de disparo de la salida digital con el potenciómetro, vertiendo agua en un pequeño experimento. Pero suele ser suficiente con dejar el potenciómetro en un valor medio.

El código es igualmente sencillo. Simplemente leemos el estado de la entrada digital. Si el sensor se dispara ejecutamos las acciones necesarias.

```
const int sensorPin = 9;

void setup() {
  Serial.begin(9600);   //iniciar puerto serie
  pinMode(pin, INPUT);  //definir pin como entrada
}

void loop(){
  int value = 0;
  value = digitalRead(sensorPin );  //lectura digital de pin

  if (value == LOW) {
     Serial.println("Detectada lluvia");
  }
  delay(1000);
}
```

VI. Sensor Receptor Infrarrojo.

Figura 52. Imagen del sensor receptor infrarrojo módulo KY-022



Fuente: Sistema Arduino

Especificación y características

Tabla 9. Características sensor receptor infrarojo

Sensor receptor infrarrojo	
Voltaje de funcionamiento: 2.7 V a 5 V	Voltaje de bajo nivel: 0.4 V
Corriente de funcionamiento: 0.4 mA a 1.5 mA	Voltaje de alto nivel: 4.5 V
Distancia de recepción: 18 m	Filtro de luz ambiente hasta: > 500 LUX
Ángulo de recepción: ± 45º	Dimensiones: 15 x 18.5 x 10 mm
Frecuencia portadora: 38 KHz	Peso: 3 gr

Fuente: Elaboración propia

El Sensor Receptor Infrarrojo IR es un módulo KY-022 que está construido de un receptor IR TL1838, el cual reacciona a la luz infrarroja de 38 KHz y funciona en conjunto con el emisor KY-005**.**

Esté módulo KY-022 se utilizan en muchos equipos domésticos, para controles remoto universales, utiliza la codificación NEC, y funciona muy bien principalmente en vehículos con MP3, marco de fotos digital, iluminación equipada.

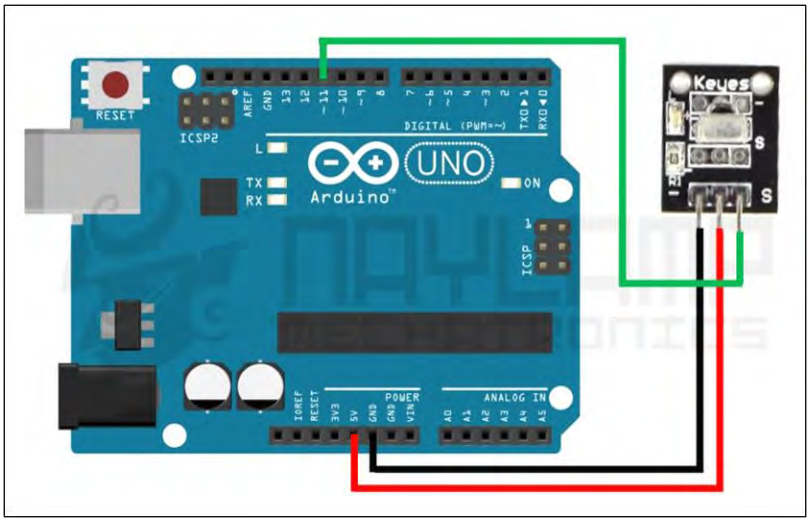
Es posible ver la luz del mando mirando el LED infrarrojo con una cámara digital, por ejemplo del celular. La luz del led se muestra como un resplandor morado. Podríamos usar este método para detectar si el mando funciona correctamente.

(Electronics U.)

Conexiones entre Arduino y modulo receptor IR

Las conexiones son simples el sensor tiene un pin VCC el cual se alimenta con 5V un pin GND y un pin de DATA, que es una salida digital el cual conectaremos al pin 11 del Arduino.

Figura 53. Ilustración de la conexión de sensor receptor infrarrojo con el sistema Arduino Uno



Fuente: Sistema Arduino

En este ejemplo se encenderá y apagará el led del pin 13 con cualquier tecla de nuestro control remoto, incluso con cualquier control.

Encendiendo un led con nuestro control Remoto.

El código es el siguiente:

```
#include <IRremote.h>

int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup()
{
  irrecv.enableIRIn(); // Empezamos la recepción  por IR
  pinMode(13, OUTPUT);
}

boolean on = LOW;

void loop() {
```

```
 if (irrecv.decode(&results)) {
   // Dato recibido, conmutamos el LED
   on = !on;
   digitalWrite(13,  on? HIGH : LOW);
   irrecv.resume(); // empezamos una nueva recepción
  }
  delay(300);
}
```

Breve Explicación el código:

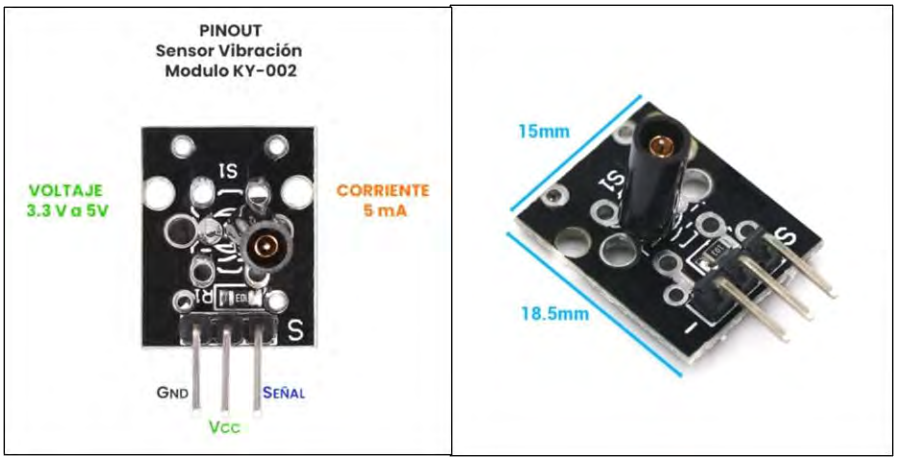
Con IRrecv (RECV_PIN) creamos la variable u objeto para el receptor IR, en el pin especificado, luego creamos la variable result, que es una estructura en donde se guardaran todos los datos relacionados cuando se recibe un dato por sensor. En Setup() inicializamos la recepción de datos con irrecv.enableIRIn() y configuramos el pin 13 como salida.

En el void loop() simplemente comprobamos si le llega un dato al receptor, esto lo hacemos con if(irrecv.decode (&results)), si hay un dato, encendemos o apagamos el LED.

Después de cargar el programa, al presionar cualquier tecla de cualquier control remoto, deberá encender o apagar el LED.

VII. Sensor de Vibración Modulo ky-002.

Figura 54. Imagen de sensor de vibración modelo KY-002



Fuente: Sistema Arduino

Especificación y características

Tabla 10. Sensor de vibración módulo KY-002

Sensor de Vibración Modulo ky-002	
Tensión de trabajo 3.3 V – 5 V	Peso: 1 g
Corriente: 5 mA	Señal digital de salida Inversa: Alta (HIGH).
Dimensiones : 18.5 mm x 15 mm	

Fuente: Elaboración propia

Significa cuando no hay detección de vibración y un nivel bajo (LOW) es porque tenemos vibración.

El módulo ky-002 es un sensor de vibración que nos permite detectar las vibraciones cuando es sometido por alguna fuerza externa. Es compatible con tarjetas Arduino y ESP8266.

Este módulo ky-002 sensor de vibración funciona como un switch el cual al detectar movimiento o una vibración manda una señal. También tenemos que considerar el posicionamiento del interruptor se debe ubicar físicamente en el área que se requiere monitorear.

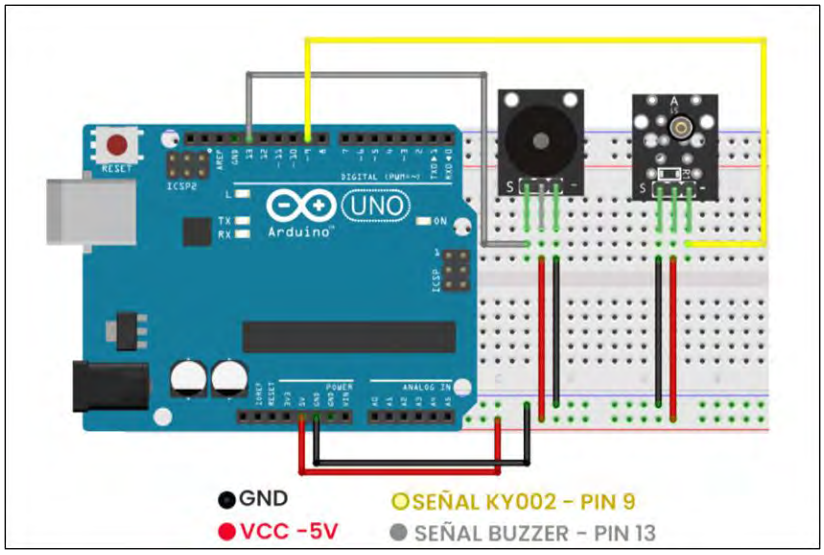
El uso del Sensor de Vibración KY-002 con el Arduino UNO. Permite detectar vibraciones o golpes de manera simple siendo funcional en proyectos para detección de sismos, sistemas de protección en motores donde la vibración puede ser prolongada y generar daños.

Se integra un zumbador pasivo, que puede reproducir tonos entre 1.5 a 2.5 KHz al encenderlo y apagarlo en diferentes frecuencias. (Electronics U. , Sensor Vibración Módulo KY-002)

Conexión del Sensor de Vibración y Arduino UNO

Las conexiones que se realizaran entre los componentes será la siguiente:

Figura 55. Conexión del sensor de vibración modelo Ky-002 con Arduino uno



Fuente: Sistema Arduino

Código en Arduino IDE para el Sensor de Vibración KY-002

El siguiente programa es útil para tener una alerta de que el sensor KY-002 está siendo perturbado. Hay que recordar que nuestro sensor de vibración tiene una lógica inversa y que al Buzzer pasivo es necesario indicarle a que frecuencia funcionara. En este caso será de 1479 Hz.

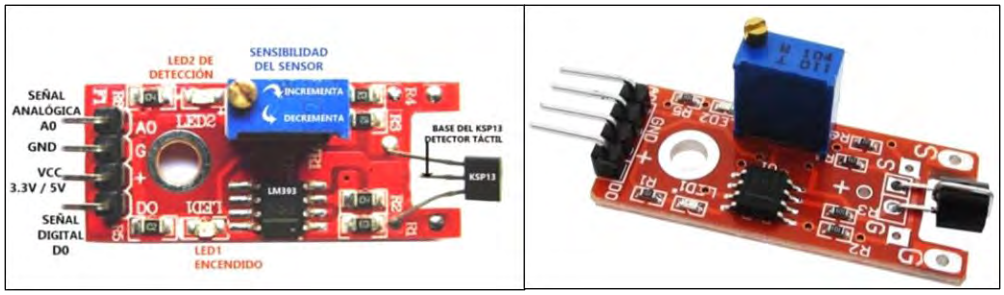
```
int buzz = 13 ;     // Definimos el pin del Buzzer, auxiliar es salida del Led13 en la PCB del Arduino
int ky02 = 9;      // Definimos el pin de detección de vibración
int alarma;        // Definimos la variable donde vamos a grabar los datos del
                   // sensor (0,1);0 si detecta movimiento y 1 sin perturbación
void setup() {

pinMode ( ky02 , INPUT ) ;// Definimos al sensor de vibración como entrada
pinMode ( buzz, OUTPUT); // Definimos al Buzzer como pin de salida
}
void loop()
{
alarma = digitalRead ( ky02) ; // Leemos el estado del pin y guardamos el valor en la variable alarma
if ( alarma == HIGH )          // Si el sensor esta sin perturbación entonces ..
{
digitalWrite ( buzz,LOW) ;       // el Buzzer no emitirá un pitido y el led L13 del Arduino NO prendera
```

```
}
Else
{
tone (buzz, 1479,1000);    //de lo contrario, si la señal es Baja y
                           //se activara el Buzzer a una frecuencia de tono 2400Hz a 1000 milisegundos
digitalWrite ( buzz,HIGH) ; / / el Buzzer  emitirá un pitido y el led L13 del Arduino prendera
}
}
```

https://blog.uelectronics.com/tarjetas-desarrollo/arduino/uso-del-sensor-de-vibracion-ky-002-como-alarma/

VIII. Sensor táctil de metal módulo KY-036.

Figura 56. Imagen sensor de metal táctil modelo KY-036 compatible con Arduino



Fuente: Sistema Arduino

Especificaciones y características

Tabla 11. Sensor táctil de metal módulo KY-036

Sensor táctil de metal módulo KY-036	
Voltaje de funcionamiento: 3.3 a 5V	*Dimensiones: 38 mm x 15 mm x 14mm*
Emite una señal al led2 si se toca la Base del sensor KSP13	*Peso: 3 g*
Interruptor digital salida (0 / 1)	*Pines*

Fuente: Elaboración propia

El Sensor Táctil de Metal módulo ky-036 es un sensor táctil que funciona con un transistor Darlington KSP13, el cual tiene una pequeña pata metálica libre que cuando es tocada

manda una señal, la cual, puede servir para cuestiones de control. La sensibilidad es ajustable con el potenciómetro.

El sensor de metal es útil para detectar cuando alguna persona lo toca y diferenciarla de muchas otras cosas, pero también es sensible a la carga de cualquier otro tipo, motivo por el cual no es recomendable su uso en aplicaciones donde se puedan presentar cargas que puedan hacer del error una variable importante.

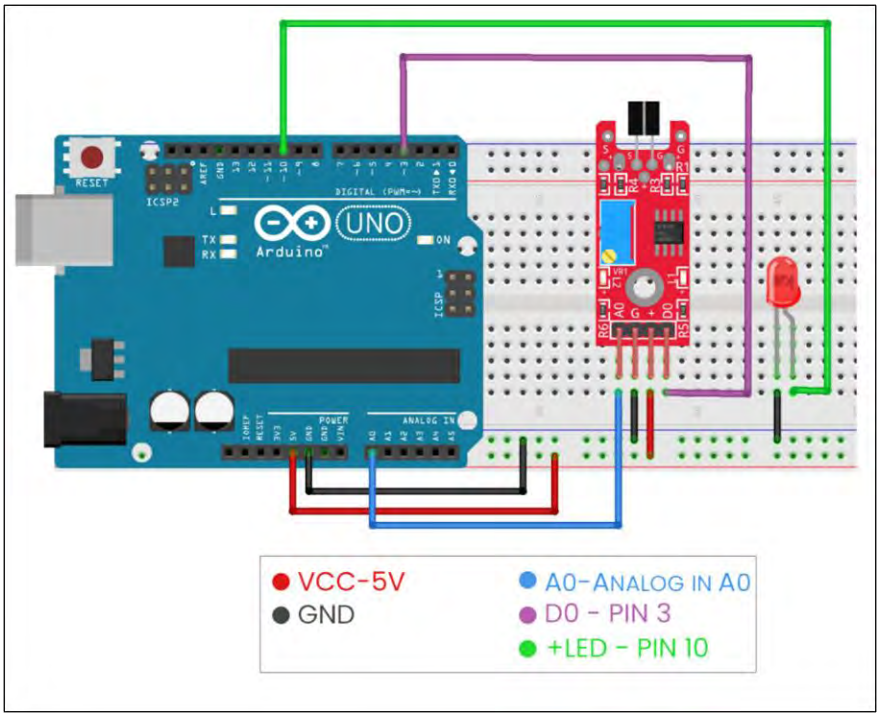
Salida digital: en el momento de la detección de contacto, se emitirá una señal

Salida analógica: valor de medición directa de la unidad del sensor

Conexiones entre KY-036 y Arduino

Las conexiones que se realizaran entre los componentes, es la siguiente:

Figura 57. Conexión del sensor de metal táctil con Arduino Uno



Fuente: Sistema Arduino

Código en Arduino IDE para el Sensor de Metal Modulo KY-036

El siguiente código nos ayudara al funcionamiento del KY-036. Encenderá el led que está conectado al pin 10 del Arduino cada vez que el sensor sea tocado. Además visualizaras por el Monitor Serie y/o Serial Plotter (en IDE Arduino) la sensibilidad del potenciómetro esto por medio de la entrada en el pin analógico A0.

```
                        //Declaración de pines y variables
int led = 10 ;              // Pin 10 para Led de salida
int touch = 3;              // Pin 3 Entrada Digital que enviara la señal cuando estemos tocando el
sensor
int potenciometro = A0;     // Pin A0 Entrada Analógica que se llamara potenciometro
int valsensor = 0;          // Variable para el potenciometro
int val = 0;                // Variable para detección del sensor
void setup ()
{
 Serial.begin (9600);           //Inicio de monitor serial a 9600 baud/seg
 pinMode (led, OUTPUT) ;        // Definimos el pin de led como salida
 pinMode (potenciometro, INPUT) ;  // Definimos potenciómetro como entrada
}
void loop ()
{
 valsensor = analogRead (potenciometro);    //se asigna a valsensor el valor de la lectura del
potenciometro
 Serial.println (valsensor);                //en el monitor serial se imprimirá el valor de lectura del
potenciometro
 val = digitalRead(touch);                  // val se le asignara el valor de salida de touch - Pin 3
 if (val == HIGH)                           //Condición , si el sensor es tocado y val es ALTO-HIGH
entonces
 {
  digitalWrite (led, HIGH);                 //prendera el led que se encuentra en la salida del pin 10
 }
 else                                       //De lo contrario , si el sensor no ha sido tocado...
 {
  digitalWrite (led, LOW);                  //no se prendera el led del pin 10
```

IX. Sensor de Flujo de Agua para Arduino.

El sensor de flujo de agua, también llamado flujómetro o caudalímetro, podemos utilizarlo en muchos proyectos con Arduino donde tengamos que mover líquidos y necesitemos medir el caudal que está pasando por nuestro sistema.

El caso típico de uso es en los sistemas de riego por aspersión o riego por goteo donde podríamos necesitar conocer cuánta agua estamos mandando a las plantas o en situaciones donde utilicemos bombas de agua y para proteger el motor necesitemos saber en todo momento si está pasando líquido por nuestra bomba.

Tipos de Caudalímetros:

Existen diferentes tipos de caudalímetros o medidores de caudal según el uso que le daremos y los recursos con los que se cuenta. Existen caudalímetros que sirven para medir distintos fluidos (agua, combustibles, aceites) y distintos volúmenes con mayor o menor precisión, y van desde los $40.00 pesos mexicanos hasta los $199,700.00 pesos mexicanos.

Presentamos diferentes tipos de sensores de flujo:

- Caudalímetro Mecánico de Turbina: Es el típico medidor de agua que tenemos todos en nuestras viviendas. El flujo hace girar un molino cuyo eje mueve un contador mecánico que acumula las lecturas. Al ser mecánico no sirve para Arduino.
- Caudalímetro Ultrasónico: Se utilizan en la industria y cuestan miles de dólares. Mide a través de cañerías de cualquier material el tiempo que tarda el ultrasonido en atravesar el fluido a medir.
- Caudalímetro Electromagnético: Utilizado en la industria para cañerías de hasta 40 pulgadas y altas presiones. También cuesta miles de dólares.
- Caudalímetro Electrónico a Turbina: Son de bajo costo y bastante precisos. La desventaja que tienen es que al ser intrusivos tienen alta perdida de carga y sufren alto deterioro en sus piezas, pero son los ideales para utilizar en los proyectos con Arduino.

El funcionamiento del sensor de flujo electrónico a turbina que se utilizara con Arduino es simple y complejo a la vez. Simple porque básicamente consta de una turbina que gira al pasar el agua a través de ella, y complejo porque utiliza un sensor de efecto Hall para calcular el flujo de agua. No es necesario describir el efecto Hall, ya que no es necesario comprenderlo para hacer un buen uso del flujómetro.

El agua circula a través del cuerpo del medidor de flujo haciendo girar la turbina o hélice que tiene dentro. Cuanto más flujo haya más rápido girarán las aspas de la turbina. Un imán situado en la turbina genera un pulso positivo cada vez que pasa por el sensor de efecto Hall. De esta forma podemos conocer las RPM que genera la hélice y calcular el caudal de agua con una sencilla ecuación.

Seguidamente le presentamos la Imagen física de un caudalímetro electrónico a turbina compatible con Arduino.

Figura 58. ilustración del caudalímetro o sensor de flujo de agua para Arduino

Fuente: Sistema Arduino

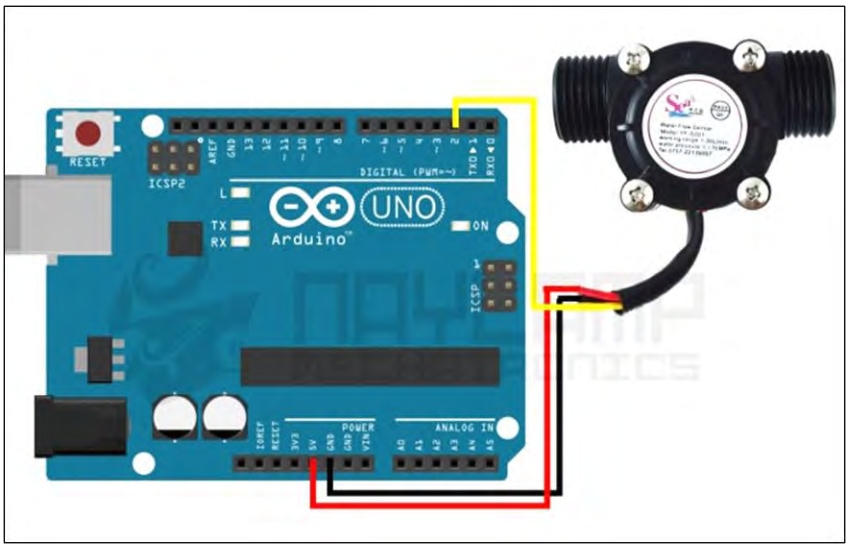
Son 4 los sensores de caudal más utilizados con Arduino, la mayor diferencia entre ellos es el tamaño de las rocas de conexión y el caudal que miden. (Arduino P. c., Sensores de Flujo de Agua para Arduino, 2018)

Conexiones entre el sensor de flujo de agua con el sistema Arduino.

La conexión eléctrica es muy sencilla, todos los sensores de flujo tienen 3 cables: rojo y negro para positivo GND y amarillo para la salida de los pulsos. Como es necesario utilizar interrupciones externas, podemos utilizar los pines 2 o 3 de Arduino uno, que son los que manejan interrupciones.

Todos los sensores de flujo tienen en la carcasa una flecha que indica el sentido en el que debe circular el líquido. Debe respetarse el sentido para que el sensor funcione correctamente y no se dañe.

Figura 59. Conexión del sensor de flujo de agua con Arduino uno



Fuente: Sistema Arduino

La conversión de frecuencia de pulsos (Hz) a caudal (L/min) varía entre modelos y depende de factores como: la presión, la densidad de líquidos e incluso el mismo caudal.

Para calcular el caudal debemos usar un factor de conversión y una formula provisto por fabricantes. Este factor de conversión solo nos servirá de referencia, ya que en la mayoría de los casos deberá ajustarse realizando las pruebas.

La fórmula: f **(Hz)= K* Q(L/min)**

Dónde: f es la frecuencia de pulsos.

K es el factor de conversión.

Q es el caudal en litros por minuto.

Despejando Q en la ecuación de la formula anterior se obtiene:

Q=f/K

Código para el sensor de flujo de agua:

En el siguiente código utilizaremos interrupciones para contar la cantidad de pulsos que se generan en un segundo y dividir el total por el factor de conversión para obtener el caudal de litros por minuto.

```
volatile int CantPulsos;        //Variable que acumula los pulsos recibidos
int LxM;                        //Variable que acumula el cálculo de Litros por Minuto
int pinsensor = 2;              //Numero de pin donde conectamos el sensor
float FacConv = 7.5;                    //Factor de Conversión para calcular caudal
void rpm ()  {                          //Función que se ejecuta durante la interrupción
  CantPulsos++;                 //Incrementa el contador de pulsos
}
void setup() {
  pinMode(pinsensor, INPUT);                    //inicializa el pin digital 2 como
entrada
  Serial.begin(9600);                           //inicializa el puerto serie
  attachInterrupt(0, rpm, RISING);      //(Interrupcion 0(Pin2),funcion rpm,Flanco de
subida, cuando pasa de LOW a HIGH)
}
void loop () {
  CantPulsos = 0;                               //Pone en 0 el contador de pulsos
  Interrupts();                         //Habilitamos las interrupciones, equivalente a
sei();
  delay (1000);                         //Esperamos un segundo
  noInterrupts();                       //Deshabilitamos las interrupciones,
equivalente a cli();
  LxM = (CantPulsos / FacConv);                 //Calcula los Litros por Minuto
  Serial.print (LxM, DEC);              //Imprime la cantidad de litros por minuto
  Serial.print (" L/minrn");            //Imprime "L/min" y salta a una nueva linea
}
```

X. Sensor de polvo para Arduino y Partículas en el Aire

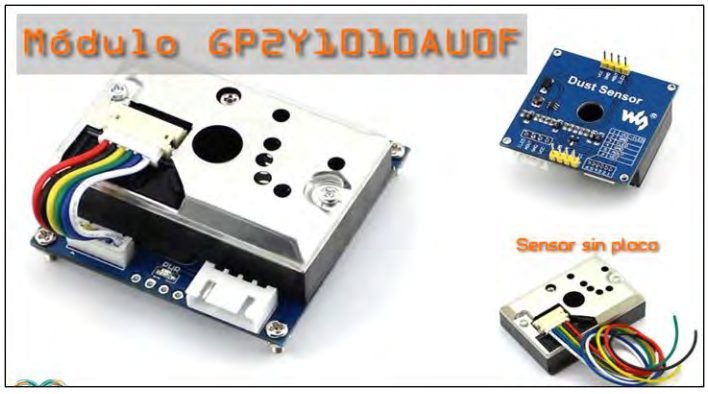
Al respirar inhalamos los gases, vapores y partículas que hay en suspensión en el aire. La exposición a estas partículas o «polvo» puede causar efectos graves sobre la salud. Existen en el mercado varios sensores de polvo que podemos utilizar con Arduino cuyo precio ronda entre los 4 y 20 dólares.

Un sensor de partículas de aire es básicamente un detector de polvo que podemos utilizar por ejemplo en alguna industria o línea de producción donde necesitemos que no haya partículas en suspensión.

Una de las características del sensor de polvo es que mide: La composición de las partículas en suspensión que inhalamos, conocidas por sus siglas en inglés PM (particulated matter), puede ser de una mezcla muy variada, pero más que por su contenido, se clasifican según su medida y según cómo se comportan al respirarlas.

La mayoría de estos sensores de polvo, según sus especificaciones técnicas, pueden detectar partículas de 2,5PM, capaces de llegar hasta los pulmones al respirarlas. La siguiente imagen 2.24 muestra el Sensor de polvo módulo GP2Y1010AU0F.

Figura 60. Sensor de polvo módulo GP2Y1010AU0F



Fuente: Sistema Arduino

Este sensor óptico desarrollado por la empresa Sharp es especialmente efectivo detectando partículas muy finas de polvo en el aire y es muy utilizado como detector en

purificadores de aire. Internamente contiene un diodo LED y un fototransistor colocado de tal forma que son capaces de detectar las reflexiones de luz sobre las partículas de polvo suspendidas en el aire que ingresa al dispositivo.

El sensor tiene un consumo de corriente muy bajo (20 mA máximo) y puede ser alimentado por hasta 7V DC. La salida del sensor es una tensión analógica proporcional a la densidad de polvo detectado.

El sensor GP2Y1010AU0F se puede adquirir integrado en una placa y listo para usar o bien sin la placa en cuyo caso se le debe añadir una resistencia de 150 ohm y un condensador de 220uF para conectarlo al Arduino. (Arduino P. c., 2018)

Especificaciones Técnicas Sensor GP2Y1010AU0F:

Tabla 12. Sensor GP2Y1010AU0F

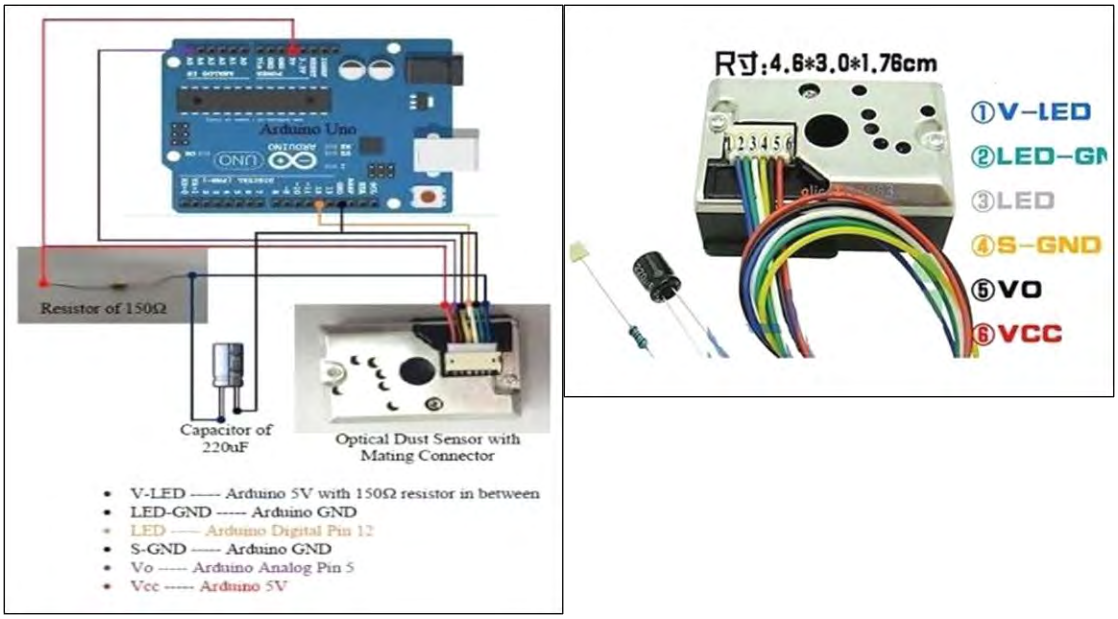
Sensor GP2Y1010AU0F:	
Sensibilidad: 0.5 V / (0.1mg/m3)	Peso: 16 g
Tensión de salida sin polvo: típica 0,9 V	Concentración de detección (typ.): 0 to 600 µg/m3.
Potencia: 7 Vdc	Temperatura a la que opera: 25- (°C)
Consumo de energía: 11 mA (máx. 20 mA)	Dimensiones: 46 x 30 x 17.6 (mm)
Temperatura de funcionamiento: -10 a +65 ° C	Bajo consumo de corriente(Icc: Max. 20mA)

Fuente: Elaboración propia

Conexiones entre el sensor de polvo módulo GP2Y1010AU0F con el sistema Arduino

Instalación y puesto en marcha del sensor de polvo con Arduino uno.

Figura 61. Conexión del sensor de polvo modelo GP2Y1010AU0F con placa o resistencia.



Fuente: Sistema Arduino

Características:

1. Delgada y compacta (46x 30 x 17.6mm).
2. La presencia de polvo puede ser detectada por la fotometría de un solo pulso.
3. Permite distinguir el uso del polvo doméstico.
4. Cumple con la directiva de ROHs y es sin plomo.
5. Detecta el polvo del aire, se emplea como purificador del aire, acondicionador del aire, monitor del aire.

Este es el código que hace funcionar el sensor de polvo detecta la densidad el polvo en el aire.

Código

```
int measurePin = 0;
int ledPower = 12;
int samplingTime = 280;

int deltaTime = 40;
int sleepTime = 9680;
float voMeasured = 0;
float calcVoltage = 0;
float dustDensity = 0;
float pm05=0;
void setup(){
Serial.begin(9600);
pinMode(ledPower,OUTPUT);
}
void loop(){
digitalWrite(ledPower,LOW); // power on the LED
delayMicroseconds(samplingTime);
voMeasured = analogRead(measurePin); // read the dust value
delayMicroseconds(deltaTime);
digitalWrite(ledPower,HIGH); // turn the LED off
delayMicroseconds(sleepTime);
// 0 - 3.3V mapped to 0 - 1023 integer values
// recover voltage
calcVoltage = 5*voMeasured/1024;
// linear eqaution taken from http://www.howmuchsnow.com/arduino/airquality/
// Chris Nafis (c) 2012
dustDensity = 0.17 * calcVoltage - 0.1;
// Ecuacion linear de PM 2.5
pm05=(calcVoltage-0.0356)*120000;
Serial.print("Raw Signal Value (0-1023): ");
Serial.print(voMeasured);
Serial.print(" - Voltage: ");
```

```
Serial.print(calcVoltage);
Serial.print(" - Dust Density(mg/m3): ");
Serial.println(dustDensity);
Serial.print(" - PM 0.5(particulas/0.01 pie3): ");
Serial.println(pm05);
delay(1000);}
```

https://proyectosconarduino.com/sensores/polvo-particulas.

XI. Sensor laser

Medir distancia con precisión con Arduino y sensor láser VL53L0X y VL6180X se puede lograr.

El VL53L0X es un sensor de distancia infrarrojo láser de última generación, que podemos emplear junto con un procesador como Arduino para medir distancias de 50mm a 2000 mm de forma precisa. Para rangos más cercanos, la variante VL6180X tiene un rango de 5mm a 200mm.

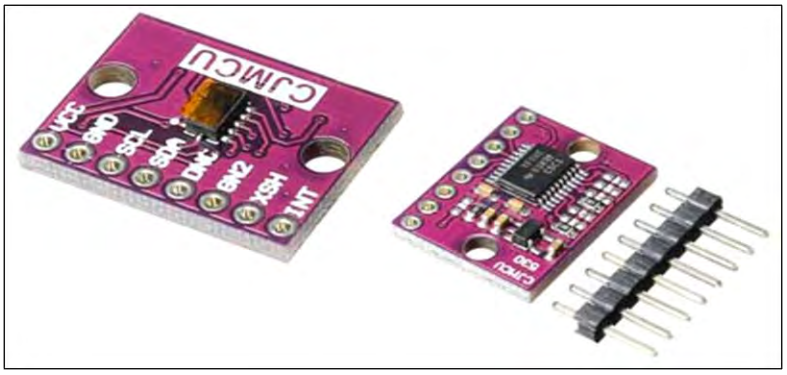
El VL53L0X es capaz de operar incluso con elevada luz ambiental infrarroja, e incorpora un sistema de compensación de la medición que lo permite hacer funcionar incluso detrás de un cristal protector.

Es uno de los mejores sensores de distancia disponibles. Tiene una precisión superior que los sensores de ultrasonidos e infrarrojos, y no se ve alterado por las condiciones del ambiente como los ecos o la reflectancia de los objetos.

Por otro lado, el ángulo de medición es relativamente estrecho. Esto es una ventaja en la mayoría de circunstancias, donde deseamos leer la distancia justo en frente del sensor. Aunque también puede ser una desventaja, por ejemplo, a la hora de detectar obstáculos.

Los sensores VL53L0X son algo más caros que otros sensores de distancia, aunque a cambio tiene unas prestaciones superiores. Podemos encontrar módulos con VL53L0X por 82.77 pesos mexicanos en vendedores internacionales de eBay y AliExpress. (Lamas, MEDIR DISTANCIA CON PRECISIÓN CON ARDUINO Y SENSOR LÁSER VL53L0X Y VL6180X, 2018|)

Figura 62. Imagen física de un sensor laser modelo VL53L0X

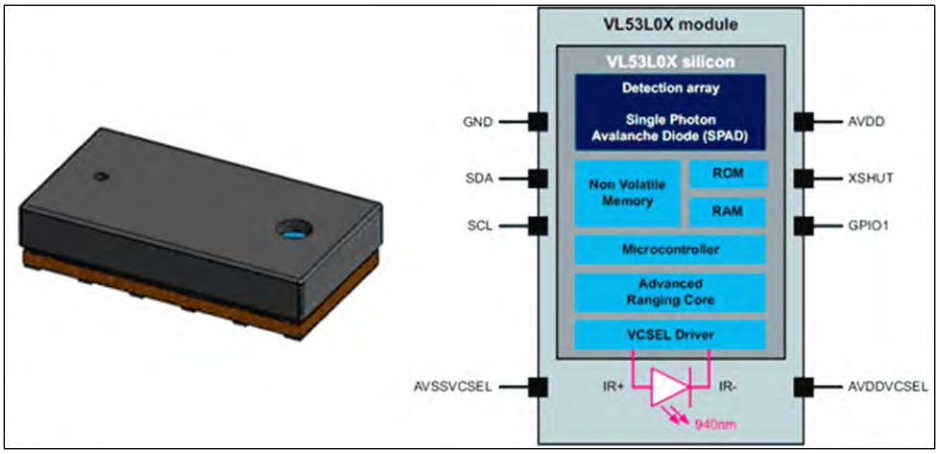


Fuente: Sistema Arduino

El VL53L0X es un sensor ToF (Time of flight). Su funcionamiento consiste en enviar un pulso láser de luz infrarroja y medir el tiempo necesario en el haz en volver al sensor.

El integrado incorpora un emisor laser 940nm VCSEL (Vertical Cavity Surface-Emitting Laser), un detector SPAD (Single Photon Avalanche Diodes) y la electrónica interna (denominada Flight SenseTM) que realiza los cálculos necesarios.

Figura 63. Estructura de la electrónica Interna sensor laser VL53L0X



Fuente: Sistema Arduino

El ángulo de medición o FOV (Field of View) es de 25º. Esto es el área de medición de 0.44m de diámetro a una distancia de 1m.

El rango de medición depende de las condiciones externas (interior, o exterior), de las características del objetivo y del modo de funcionamiento. En general, tenemos dos modos. El estándar es de 50 a 1200mm, y un modo ampliado hasta 2000mm. Para rangos más cercanos, la variante VL6180X tiene un rango de 5mm a 200mm.

En cuanto a precisión, nuevamente depende del entorno, objetivo, y modo de funcionamiento.

La siguiente tabla muestra valores típicos de rango y precisión según los distintos formas de funcionamiento del VL53L0X.

Tabla 13. Características sensor laser VL53L0X

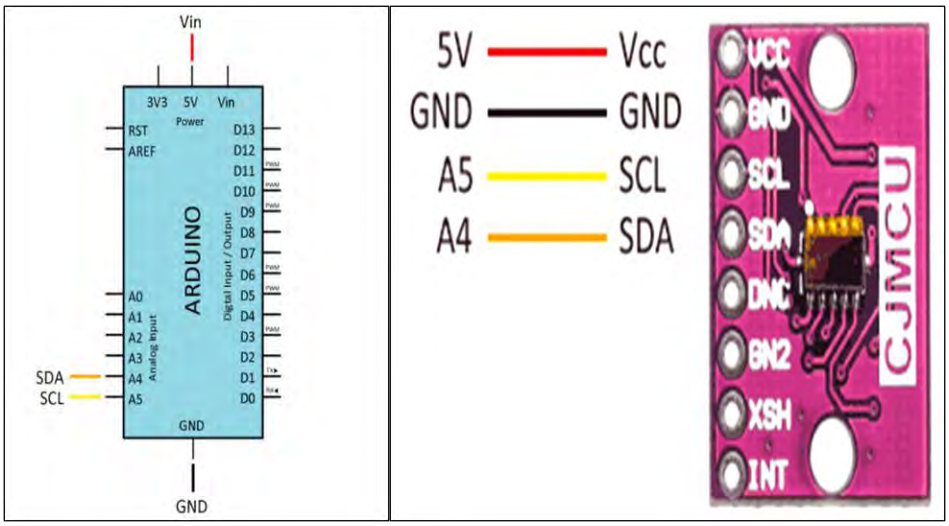
Modo sincronización	***sincronización***	***Alcance***	***Precisión***
Default	30ms	1.2m	ver tabla siguiente
Alta precisión	200ms	2m	+/- 3%
Largo alcance	33ms	1.2m	Ver tabla siguiente
Alta velocidad	20ms	1.2m	+/- 5%

Fuente: Elaboración propia

Esquema de Montaje

La conexión de los módulos que integran el VL53L0X es sencilla, ya que la comunicación se realiza a través de I2C. Por tanto, simplemente alimentamos el módulo desde Arduino mediante GND y 5V y conectamos el pin SDA y SCL de Arduino con los pines correspondientes del sensor.

Figura 64. Conexión del sensor laser VL53L0X con Arduino



Fuente: Sistema Arduino

La librería incorpora varios ejemplos de uso. El siguiente código toma lecturas de la distancia y muestra los resultados por puerto serie.

```
#include "Adafruit_VL53L0X.h"
Adafruit_VL53L0X lox = Adafruit_VL53L0X();
void setup() {
 Serial.begin(9600);
// Iniciar sensor
Serial.println("VL53L0X test");
  if (!lox.begin()) {
 Serial.println(F("Error al iniciar VL53L0X"));
 while(1);
  }
}
void loop() {
  VL53L0X_RangingMeasurementData_t measure;
    Serial.print("Leyendo sensor... ");
lox.rangingTest(&measure, false); // si se pasa true como parametro, muestra por puerto serie datos de debug
 if (measure.RangeStatus != 4)
```

```
 {
   Serial.print("Distancia (mm): ");
Serial.println(measure.RangeMilliMeter);
 }
 Else
 {
 Serial.println("  Fuera de rango ");
}delay(100);
}
```

www.luisllamas.es/arduino-sensor-distancia-vl53l0x/

XII. Sensor de Temperatura y Humedad.

Medir temperatura y humedad con Arduino y sensores DHT11 y el DHT22 (o AM2302) son dos modelos de una misma familia de sensores, que permiten realizar la medición simultánea de temperatura y humedad.

Estos sensores disponen de un procesador interno que realiza el proceso de medición, proporcionando la medición mediante una señal digital, por lo que resulta muy sencillo obtener la medición desde un microprocesador como Arduino.

Ambos sensores presentan un encapsulado de plástico similar. Podemos distinguir ambos modelos por el color del mismo. El DHT11 presenta una carcasa azul, mientras que en el caso del sensor DHT22 el exterior es blanco.

De ambos modelos, el DHT11 es el hermano pequeño de la familia, y cuenta peores características técnicas. El DHT22 es el modelo superior pero, por contra, tiene un precio superior.

Las características del DHT11 son realmente escasas, especialmente en rango de medición y precisión.

Medición de temperatura entre 0 a 50, con una precisión de 2ºC

Medición de humedad entre 20 a 80%, con precisión del 5%.

Frecuencia de muestreo de 1 muestras por segundo (1 Hz)

El DHT11 es un sensor muy limitado que podemos usar con fines de formación, pruebas, o en proyectos que realmente no requieran una medición precisa.

Por el contrario, el modelo DHT22 tiene unas características mucho más aceptables.

Medición de temperatura entre -40 a 125, con una precisión de 0.5ºC

Medición de humedad entre 0 a 100%, con precisión del 2-5%.

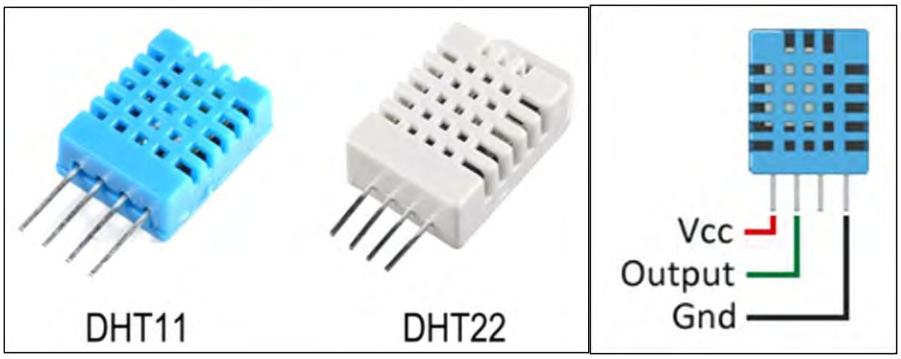
Frecuencia de muestreo de 2 muestras por segundo (2 Hz)

EL DHT22 sensor que no cuenta con alta precisión aun que es aceptables para emplearlo en proyectos reales de monitorización o registro, que requieran una precisión media.

El DHT11, el modelo inferior, es realmente barato. Podemos encontrarlo en vendedores internacionales en Ebay y AliExpress por $15.00 pesos mexicanos.

Mientras, podemos encontrar el modelo superior DHT22 por $51.00 pesos mexicano, considerablemente más caro que DHT11, pero aun relativamente barato. (Llamas, MEDIR TEMPERATURA Y HUMEDAD CON ARDUINO Y SENSOR DHT11-DHT22, 2016)

Figura 65. Sensor de medición de temperatura y humedad con su diagrama de conexión

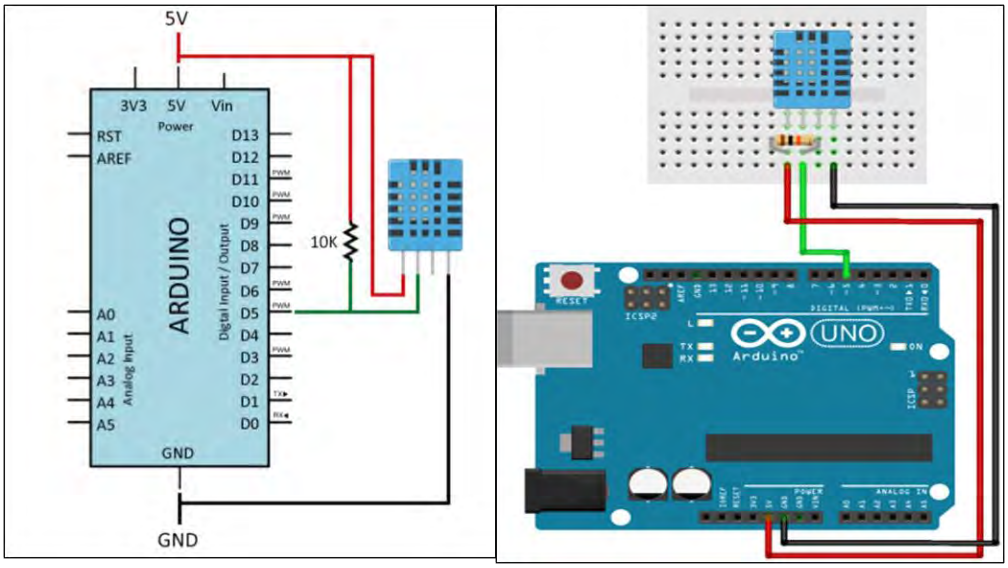


Fuente: Sistema Arduino

Conectar el sensor es sencillo, simplemente alimentamos desde Arduino al sensor a través de los pines GND y Vcc del mismo. Por otro lado, conectamos la salida Output a una entrada digital de Arduino. Necesitaremos poner una resistencia de 10K entre Vcc y el Pin Output.

El esquema eléctrico queda como la siguiente imagen:

Figura 66. Imagen de la conexión del Arduino con el sensor DHT11para medir temperatura y humedad.



Fuente: Sistema Arduino

Los sensores DHT11 y DHT22 usan su propio sistema de comunicación bidireccional mediante un único conductor, empleando señales temporizadas.

En cada envío de medición el sensor envía un total de 40bits, en 4ms. Estos 40 bits corresponden con 2 Bytes para la medición de humedad, 2 Bytes para la medición de temperatura, más un Byte final para la comprobación de errores (8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check sum)

Existen varias librerías disponibles. Por ejemplo, podemos usar la librería de Adafruit disponible en este enlace.

Descargamos e instalamos la librería y cargamos el código de ejemplo, o la siguiente versión simplificada

Código de funcionamiento los sensores DHT11 y DHT22

```
#include "DHT.h"
/ Uncomment whatever type you're using!
//#define DHTTYPE DHT11   // DHT 11
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
//#define DHTTYPE DHT21   // DHT 21 (AM2301)
// Connect pin 1 (on the left) of the sensor to +5V
// NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1
// to 3.3V instead of 5V!
// Connect pin 2 of the sensor to whatever your DHTPIN is
// Connect pin 4 (on the right) of the sensor to GROUND
// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor
const int DHTPin = 5;     // what digital pin we're connected to
DHT dht(DHTPin, DHTTYPE);
void setup() {
  Serial.begin(9600);
  Serial.println("DHTxx test!");
  dht.begin();
}
void loop() {
// Wait a few seconds between measurements.
delay(2000);
// Reading temperature or humidity takes about 250 milliseconds!
float h = dht.readHumidity();
 float t = dht.readTemperature();
if (isnan(h) || isnan(t)) {
 Serial.println("Failed to read from DHT sensor!");
 return;
}
Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");
```

XIII. Sensor de sonido módulo KY-037.

Figura 67. Sensor de sonido módulo KY-037.

Fuente: Sistema Arduino

Figura 68. Imagen del sensor KY-037

Módulo	Caracteristicas	Sensibilidad
KY-037 Sensor de Sonido 	Detección de sonidos por medio de un microfono, que varia dependiento el ajuste de sensibilidad del potenciometro.	Mayor sensibilidad
KY-038 Sensor Micrófono 	Cuando se detecte el sonido otorgará a una salida **ALTA (analógica o digital)**,esto solo si supera el umbral ajustado por el potenciometro.	Sensibilidad Normal

Fuente: Sistema Arduino

El Módulo Ky-037 Sensor de Sonido permite detectar cualquier tipo de sonido. Incluye un trimpot con el cual se puede ajustar la sensibilidad del sensor la información de salida puede ser analógica y/o digital.

El sensor de sonido es útil para para encender o apagar alguna lámpara, para detector de ruido en algún lugar de trabajo u hogar. El costo es muy económico en $21 pesos mexicanos.

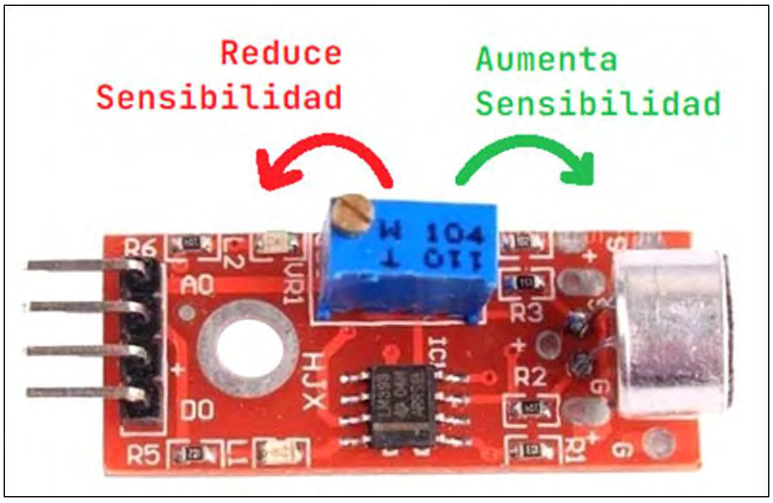
Especificaciones y características:

- Voltaje de funcionamiento: 5 V
- Salidas: Analógica y digital
- Permite ajustar un nivel de umbral de salida
- Usa el Micrófono Gao Gan grado, de alta sensibilidad.
- Interruptor digital salida (0 / 1)
- Temperatura: -40 a +85 °C
- Dimensiones: 35 x 15 x 14 mm
- Peso: 4 g

Con los sensores de sonido KY-038 y KY-037 podemos controlar el encendido y apagado de un LED; y posteriormente de un foco, utilizando un módulo relevador de 5V, KY-019.

Los sensores de sonido KY-038 y KY-037 son muy similares, ya que ambos cuentan con un micrófono para la detección de sonidos, la única diferencia es la sensibilidad que tiene cada uno, el ajuste de la sensibilidad es a través del trimpot, siendo que el giro sea en la dirección de las manecillas del reloj aumenta la sensibilidad y del lado contrario se reduce. (Eectronic, https://uelectronics.com/producto/modulo-ky-037-sensor-de-sonido/)

Figura 69. Ilustración del ajuste de la sensibilidad del sensor KY-037b o KY038.



Fuente: Sistema Arduino

Los dispositivos tienen dos Leds:

- **L1:** Confirma que el dispositivo esta alimentado (3 a 5 V).
- **L2:** Detección de sonido, si enciende; entonces se ha detectado el sonido en el rango ajustado.

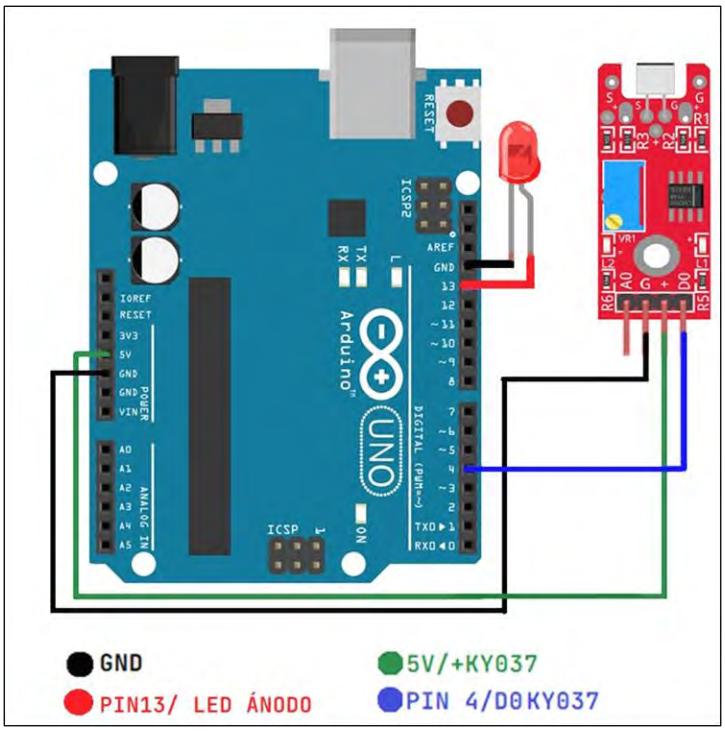
Ahora para realizar la prueba de funcionamiento y ajuste de sensibilidad de ambos sensores, lo cual se requiere:

- Sensor Micrófono KY-038
- LED de 5 mm
- Sensor de Sonido KY-037
- Arduino UNO R3
- Cables Dupont

Diagrama de conexión de los sensores de sonido

La siguiente conexión es útil para ambos sensores:KY-037 y KY-038, ya que ambos cuentan con los mismos pines. El led que estaremos conectando nos confirmará la detección de sonido.

Figura 70. Diagrama de conexión de Arduino uno con el sensor KY-038 o KY-037



Fuente: Sistema Arduino

Código de prueba de Sensores de Sonido

El siguiente programa tiene como función detectar los pulsos (que llamaremos flancos) que está recibiendo la tarjeta de desarrollo cada vez que detecte sonido.

/Nota: Ajustar la sensibilidad del micrófono con el trimpot

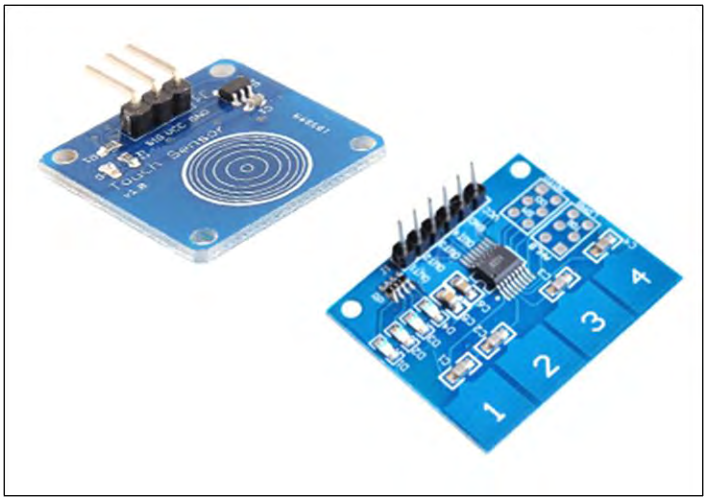
```
int KY037= 4;    //Pin 4 para la salida digital del KY
int flanco= 0;   //Variable de apoyo para saber la lectura del sensor
```

```
void setup()
{
  Serial.begin(9600);      //Inicialización del puerto serial a 9600 baudios
  pinMode( KY037, INPUT) ; //pin del KY como entrada de datos
}

void loop()
{
  if (digitalRead(KY037)==HIGH )          //Si se detecta un sonido el KY(dependiente a la sensibilidad determinada por el usuario)
   {
    flanco= flanco + 1;                   // Se realiza un contador +1 por cada vez que el sensor detecte un flanco ALTO
    Serial.print("Número de Flancos: "); // Escribe el número de flancos detectados dependiendo el sonido censado...
    Serial.println(flanco, DEC) ;         // escribiendo el valor en forma decimal
   }
}
```

https://blog.uelectronics.com/tarjetas-desarrollo/uso-de-los-sensores-de-sonido-ky-038-ky-037.

XIV. Sensor táctil de metal.

Figura 71. Imagen del sensor táctil o sensor touchless



Fuente: Sistema Arduino

Un sensor táctil capacitivo es un dispositivo que presenta un comportamiento similar a un pulsador y puede ser activado con poca o ninguna presión.

Este tipo de sensor táctil basa su funcionamiento en la medición de la variación de la capacitancia**.** La placa sensora y el cuerpo humano actúan como condensador y, por tanto, forman un sistema que almacena una carga electricidad.

Al reducir la distancia la capacitancia aumenta y el sistema almacena una carga superior. Esta acumulación de carga puede ser detectada en la placa censora y generar una señal digital cuando supere un cierto valor. Esta señal de disparo puede a ser a su vez ser capturada con una entrada digital de Arduino.

La principal ventaja de este tipo de sensores es que no requieren de contacto físico para realizar el disparo**,** siendo suficiente acercar el dedo a 1-5mm del sensor. Por este motivo se les denomina dispositivos touchless.

De esta forma, es posible colocar el sensor táctil debajo de un vinilo, plástico, cartón, madera o cristal, siempre que el espesor no sea excesivo. Por otro lado, no funcionarán debajo de materiales conductores, en particular debajo de metales.

Otra ventaja de los sensores capacitivos es que carecen de partes móviles por lo que, en principio, tienen una durabilidad superior a la de un interruptor convencional.

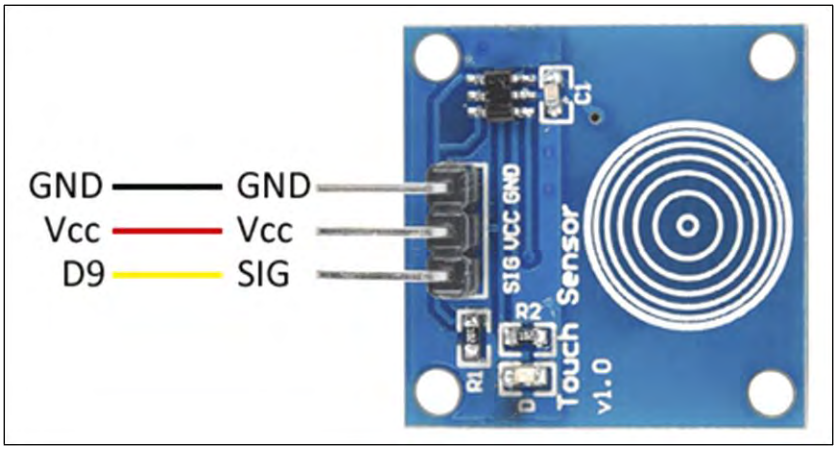
Los sensores touchless son empleados a la hora de hacer interruptores eléctricos, por ejemplo, son frecuentes en baños y garajes. También pueden ser útiles, por ejemplo, para ubicar un pulsador táctil bajo un panel interactivo, de un vinilo con artes gráficas, o integrado bajo la madera de un mueble.

Este tipo de sensores táctiles capacitivos son dispositivos baratos. Existen placas integradas listas para conectar a Arduino, con distintos tamaños, formas, y número de contactos. Podemos encontrar un sensor táctil de 1 pulsador por $8.50 Moneda Nacional, el de 1x4 pulsadores por $13.60 Moneda Nacional, y el de 4x4 pulsadores por $20.94, en vendedores internacionales de eBay y AliExpress.

Esquema de montaje

El esquema eléctrico es sencillo. Alimentamos el módulo conectando GND y 5V a los pines correspondientes de Arduino.

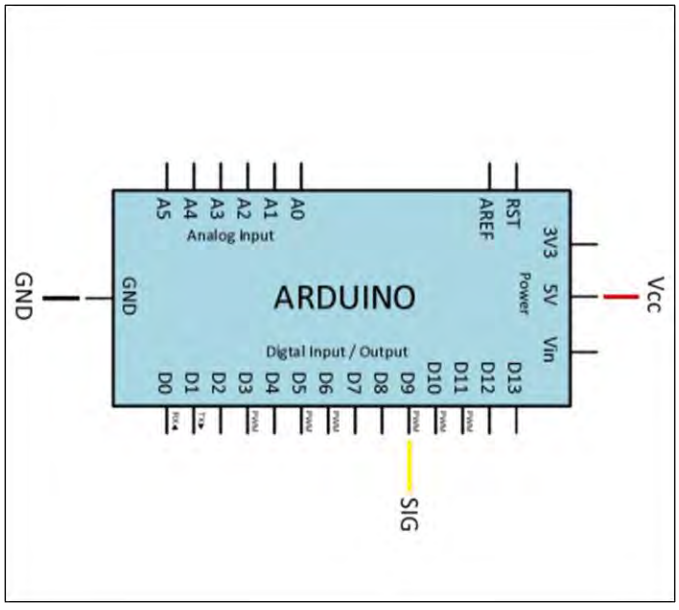
Figura 72. Diagrama de conexión del sensor táctil o touchless.



Fuente: Sistema Arduino

Finalmente, para realizar la lectura digital conectamos la salida SIG a una de las entradas digitales de Arduino.

Figura 73. Conexión de los pines correspondientes del Arduino con l sensor táctil.



Fuente: Sistema Arduino

Ejemplos de código

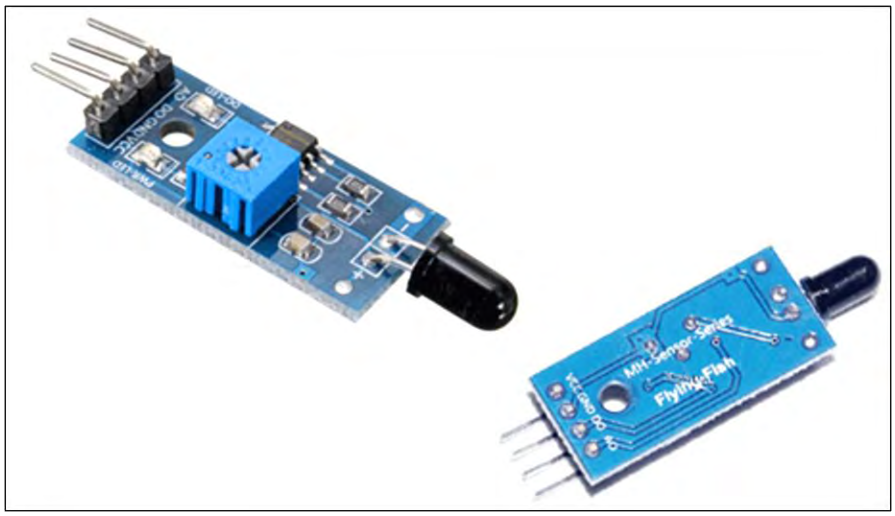
El código es igualmente sencillo. Si estamos empleando la señal digital, se emplea una entrada digital para leer el estado. En el siguiente ejemplo mostramos un mensaje por la pantalla, pero en un caso real ejecutaríamos las acciones oportunas.

```
const int sensorPin = 9;

void setup()
{
  Serial.begin(9600);
  pinMode(sensorPin, INPUT);
}

void loop()
{
  int estado = digitalRead(sensorPin);

  //mandar mensaje a puerto serie en función del valor leido
  if (estado == HIGH)
  {
 Serial.println("Contacto detectado");
    //aquí se ejecutarían las acciones
  }
  delay(1000);
}
```

XV. Sensor de llama o sensor de llama infrarrojo

Figura 74. Imagen del sensor de llama o sensor infrarrojo

.



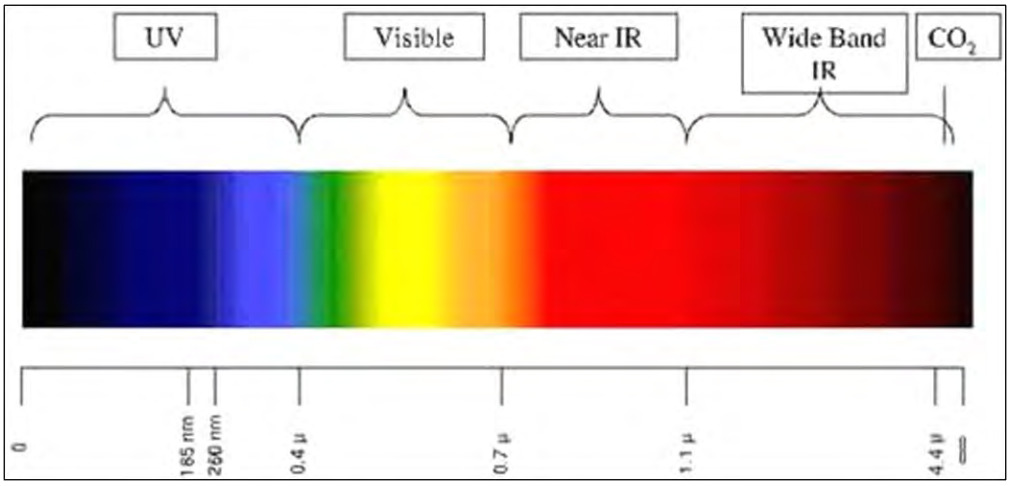
Fuente: Sistema Arduino

Un sensor de llama óptico es un dispositivo que permite detectar la existencia de combustión por la luz emitida por la misma. Esta luz puede ser detectada por un sensor óptico, y ser capturado por las entradas digitales y las entradas analógicas de Arduino.

La llama es un fenómeno de emisión de luz asociado a los procesos de combustión. La combustión es un proceso que desprende grandes cantidades de energía en forma de calor. Durante la reacción se generan compuestos intermedios que liberan parte de su energía mediante la emisión de luz.

El espectro de emisión de llama depende de los elementos que intervienen en la reacción. En el caso de combustión de productos con carbón en presencia del oxígeno tenemos dos picos característicos en ultravioleta en longitudes de onda de 185nm-260nm y en infrarrojo en longitudes de onda 4400-4600nm.

Figura 75. Espectro de emisión de llama.



Fuente: Sistema Arduino

Por este motivo se incorporan sensores de llaman como dispositivo de seguridad, permitiendo detener el proceso en caso de detectar cualquier indicio de combustión. Estos dispositivos se ajustan a las longitudes de onda características de la aparición de la llama y normalmente combinan las señales ultravioletas y de infrarrojo.

En el campo de los hobbies podemos encontrar sensores de llama baratos. Frecuentemente estos sensores constan únicamente de un sensor infrarrojo ajustado a 760-1100nm. El ángulo de detección es de 60º, y la distancia de detección entre 0.40 m a 0.80m.

Este tipo de sensores de llama infrarrojos suelen incorporar una placa de medición estándar con el comparador LM393, que permite obtener la lectura tanto como un valor analógico como de forma digital cuando se supera un cierto umbral, que se regula a través de un potenciómetro ubicado en la placa.

Longitudes de onda de estos sensores baratos poco tienen que ver con las emisiones características de las llamas y de los sensores industriales. De hecho, son afectados incluso por la iluminación interior, dando lugar a numerosos falsos positivos.

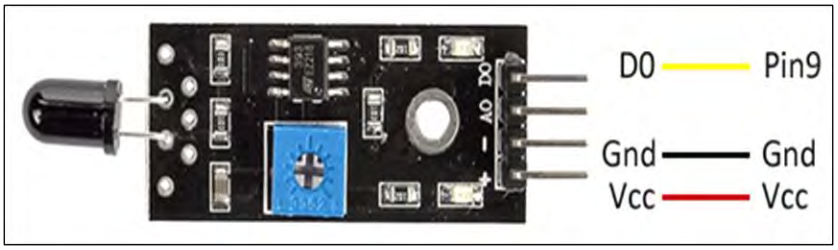
Por tanto, la sensibilidad y fiabilidad de estos detectores baratos no son suficientes para considerarlos un auténtico dispositivo de seguridad, aunque pueden ser interesantes en pequeños proyectos de electrónica y con fines didácticos, como por ejemplo hacer sonar una alarma o activando un LED al detectar la llama de un mechero.

Los sensores de llama infrarrojos caseros son dispositivos baratos. Podemos encontrarlos, incluida la placa de medición, por 0,65€ en vendedores internacionales de Ebay y Aliexpress.

Esquema de montaje

El esquema eléctrico es sencillo. Alimentamos el módulo conectando GND y 5V a los pines correspondientes de Arduino. Ahora si queremos usar la lectura digital, conectamos la salida DO a una de las entradas digitales de Arduino.

Figura 76. Terminales de conexión del sensor de llama con el Arduino



Fuente: Sistema Arduino

Mientras que la conexión vista desde Arduino quedaría de la siguiente manera:

Figura 77. Ilustración de los pines de Arduino con el sensor de llama o infrarrojo.

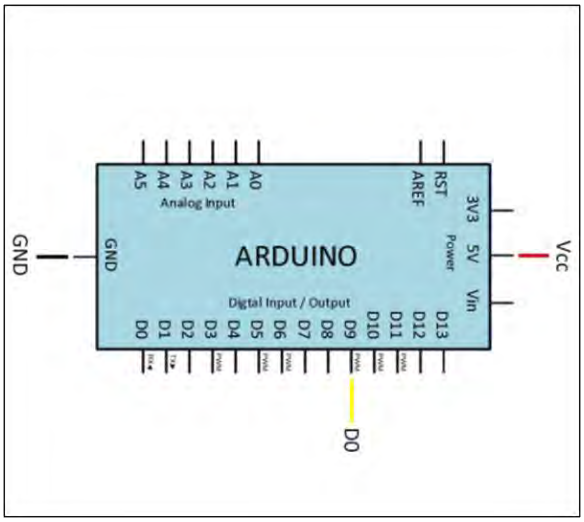


Figura: Sistema Arduino

Si quisiéramos emplear el valor analógico, simplemente conectaríamos la salida AO del sensor a una entrada analógica de Arduino.

Ejemplos de código

El código necesario es sencillo. Si estamos empleando la señal digital, empleamos una entrada digital para leer el estado. En el ejemplo mostramos un mensaje por la pantalla, pero en un caso real ejecutaríamos las acciones oportunas.

```
const int sensorPin = 9;

void setup()
{
  Serial.begin(9600);
  pinMode(sensorPin, INPUT);
}

void loop()
{
  int humedad = digitalRead(sensorPin);

  //mandar mensaje a puerto serie en función del valor leido
  if (humedad == HIGH)
  {
    Serial.println("Detección");
    //aquí se ejecutarían las acciones
  }
 delay(1000);
}
```

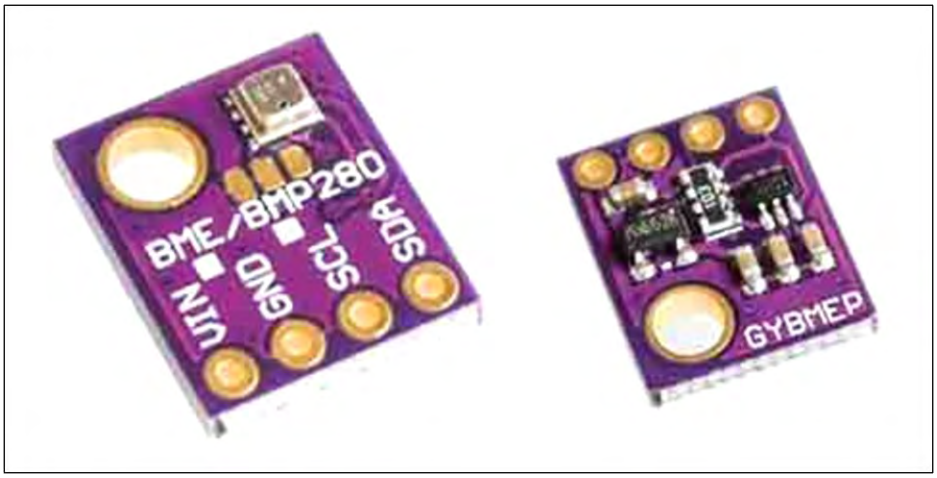
Si estamos empleando la señal analógica AO, leemos el valor mediante la entrada analógica, y usamos el puerto serie para mostrar el valor por pantalla. En un caso real, este valor se emplearía para ejecutar acciones, en lugar de mostrar el valor.

```
const int sensorPin = A0;
void setup() {
  Serial.begin(9600);
```

```
}
void loop()
{
 int measure = analogRead(sensorPin);
  Serial.println(measure);
  if(measure < 500)
   {
     Serial.println("Detección");
     //hacer las acciones necesarias
  }
  delay(1000);
}
```

XVI. Sensor BME280 compatible con Arduino uno.

Figura 78. Sensor modelo BME280 mide temperatura y presión



Fuente: Sistema Arduino

Medir temperatura y presión barométrica con arduino y bmp280

El BMP280 es un termómetro y barómetro digital del fabricante Bosch Sensortech, que permite realizar lecturas de temperatura, presión atmosférica, y estimación de altitud sobre el nivel del mar.

El BMP280 es un reemplazo de mayor precisión, menor consumo y menor tamaño, a la vez que mantiene su bajo coste, incluso inferior a los modelos anteriores.

Los rangos de medición para temperatura el rango es de es de -40º a 85ºC, con una precisión de ±1.0C.

Para presión atmosférica / altímetro es de 300hPa a 1110 hPa, equivalente a una altitud de -500m a 9000m sobre el nivel del mar. Con una precisión absoluta de ±1.0C y de 1.0 hPa, y la relativa de 0.12 hPa, equivalente a una precisión en altitud de aproximadamente ±1m.

Por su parte, la resolución de la medición mejora sustancialmente, pasando a 0.01ºC (desde 0.1ºC en el BMP180) en mediciones de temperatura, y 0.16 Pa (desde 1 Pa en el BMP180).

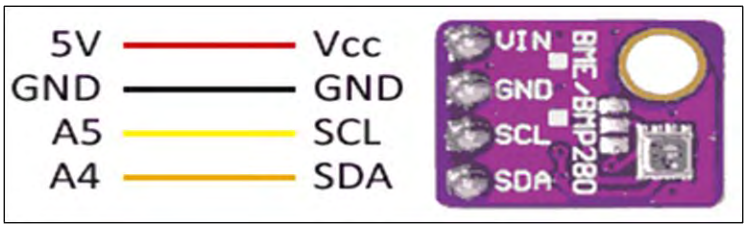
Por otro lado, el tamaño del BMP280 es un 63% más pequeño que el BMP180. El consumo también mejora, pasando de 12 µA 2.7 µA para mediciones a 1Hz.

El BMP280 incorpora comunicación I2C y SPI, por lo que es muy sencillo conectarlo a un procesador como Arduino. En comparación, el modelo anterior sólo disponía de I2C.

La tensión de alimentación es 1.8 a 3.6V, y es posible encontrarlo tanto en módulos de 3V3 como en módulos de 5V, que incorporan la adaptación de nivel necesaria.

El BMP280 es un sensor bastante barato, si consideramos que dispone de medición de temperatura, humedad y presión en el mismo dispositivo. El precio suele ser menor en los módulos de 3V3, respecto a los que permiten tanto 3V3/5V. El BMP280 compatible 3V3/5V por unos $25.34 pesos mexicanos, en vendedores internacionales en eBay o AliExpress.

Figura 79. Sensor BMP280 diagrama de conexión



Fuente: Sistema Arduino

Mientras que la conexión vista desde el lado de Arduino quedaría así.

La dirección del bus cambia según el estado lógico del pin SDO, y si se deja desconectado la dirección queda indeterminada, por lo que puede parecer que no funciona correctamente.

Ejemplos de código

Para realizar la lectura del BMP280 usaremos la librería desarrollada por Adafruit, el bus I2C utiliza por defecto la dirección (0x77), para modificarlo hay que editar el fichero "Adafruit_BMP280.h" y en la línea 37 #define BMP280_ADDRESS (0x77) cambiamos la dirección (0x76).

La librería proporciona ejemplos de código, que resulta aconsejable revisar. Los siguientes ejemplos, por ejemplo, son modificaciones están basados a partir de los disponibles en la librería.

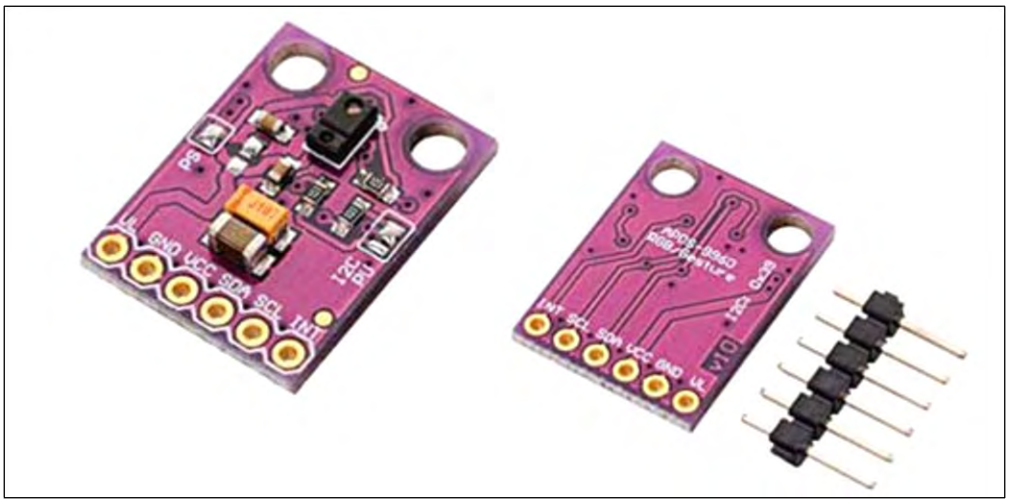
Para obtener los valores de temperatura, humedad y presión, este ejemplo se obtiene la medición de los valores en bruto (RAW) de temperatura, humedad y presión, y los muestra en la pantalla. Estos valores son de utilidad, por ejemplo, para hacer una estación meteorológica.

```
Adafruit_BMP280 bmp;
void setup() {
Serial.begin(9600);
Serial.println(F("BMP280 test"));
if (!bmp.begin()) {
Serial.println(F("Could not find a valid BMP280 sensor, check wiring!"));
while (1);
}
bmp.setSampling(Adafruit_BMP280::MODE_NORMAL, /* Modo de operación */
Adafruit_BMP280::SAMPLING_X2, /* Temp. oversampling */
Adafruit_BMP280::SAMPLING_X16, /* Presion oversampling */
Adafruit_BMP280::FILTER_X16, /* Filtrado. */
Adafruit_BMP280::STANDBY_MS_500); /* Tiempo Standby. */
}
void loop()
```

```
{
Serial.print(F("Temperatura = "));
Serial.print(bmp.readTemperature());
Serial.println(" *C");
Serial.print(F("Presión = "));
Serial.print(bmp.readPressure());
Serial.println(" Pa");
Serial.print(F("Altitud = "));
Serial.print(bmp.readAltitude(1013.25));
Serial.println(" m");
Serial.println();
delay(2000);
}
```

XVII. Detectar gestos con Arduino y sensor APDS-9960

Figura 80. Sensor APDS-9960



Fuente: Sistema Arduino

El APDS-9960 integra un emisor de infrarrojos, cuatro fotodiodos direccionales y cuatro fotodiodos para la medición de color. Además, incorpora la electrónica necesaria para la medición, con rechazo de luz ambiental y compensción de offset.

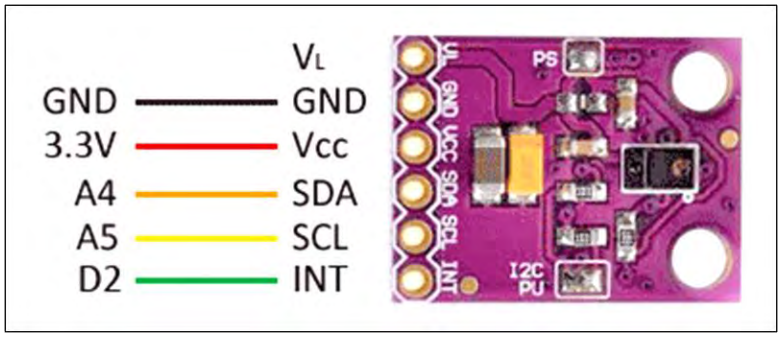
Para la detección de movimientos, el APDS-9960 compara las variaciones de luz registrada en cada uno de los fotodiodos, lo que le permite estimar el movimiento frente al sensor a una distancia de entre 5-20cm.

Para la detección de proximidad, el APDS-9960 mide la cantidad total de luz infrarroja registrada, lo que permite estimar la distancia a la que está un objeto frente al sensor. (Llamas, DETECTAR GESTOS CON ARDUINO Y SENSOR APDS-9960, 2018)

Esquema de montaje

La conexión de los módulos que integran el APDS-9960 es sencilla, ya que la comunicación se realiza a través de I2C. Sin embargo, la tensión de alimentación del APDS-9960 es de 3.3V. Por otro lado, para la comunicación por I2C debemos conectar los pines SCK y SDA con los pines correspondientes de nuestro modelo de Arduino. Los pines de comunicación deben funcionar igualmente a 3.3V. Si nuestro modelo de Arduino es de 5V, debemos usar un adaptador de nivel lógico para realizar la conexión del bus I2C. No obstante, en los experimentos con comunicación a 5V el APDS-9960 funciona correctamente, pero nos arriesgamos a dañar el módulo o reducir su vida útil.

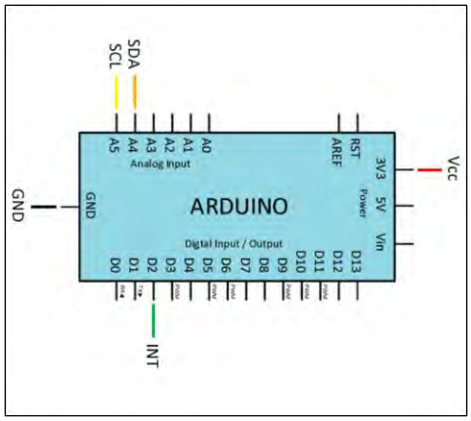
Figura 81. Diagrama de conexión del sensor APDS-996



Fuente: Sistema Arduino

La conexión, vista desde Arduino, sería la siguiente.

Figura 82. Conexión del Arduino con el sensor APDS-9960.



Fuente: Sistema Arduino

Ejemplos de código

Para realizar la lectura del sensor se utilizará la librería desarrollada por Adafruit, disponible en este enlace.

A continuación, vamos a ver algunos ejemplos de uso del sensor APDS-9960.

En este primer ejemplo, leemos los gestos y mostramos por puerto serie el resultado. Para probarlo, pasar la mano por encima del sensor a unos 4-20cm de distancia.

```
#include "Adafruit_APDS9960.h"
Adafruit_APDS9960 apds;
void setup() {
  Serial.begin(115200);

  if(!apds.begin()) Serial.println("failed to initialize device! Please check your wiring.");
  apds.enableProximity(true);
  apds.enableGesture(true);
}
void loop() {
    uint8_t gesture = apds.readGesture();
    if(gesture == APDS9960_DOWN) Serial.println("DOWN");
    if(gesture == APDS9960_UP) Serial.println("UP");
    if(gesture == APDS9960_LEFT) Serial.println("LEFT");
    if(gesture == APDS9960_RIGHT) Serial.println("RIGHT");
```

Mostrar por led integrado

En este ejemplo, hacemos parpadear el LED integrado en la placa un número de veces, en función del gesto detectado. Esto nos permite comprobar el correcto funcionamiento del sensor sin necesidad de un ordenador para visualizar la salida.

```
#include "Adafruit_APDS9960.h"
Adafruit_APDS9960 apds;

void setup()
{
 Serial.begin(115200);

 pinMode(LED_BUILTIN, OUTPUT);

 if(!apds.begin()) Serial.println("failed to initialize device! Please check your wiring.");
 apds.enableProximity(true);
 apds.enableGesture(true);
}

void loop()
{
   uint8_t gesture = apds.readGesture();
   if(gesture == APDS9960_DOWN) blink(1);
   if(gesture == APDS9960_UP) blink(2);
   if(gesture == APDS9960_LEFT) blink(3);
   if(gesture == APDS9960_RIGHT) blink(4);
}

void blink(int count)
{
 for(int i = 0; i < count; i++)
 {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(100);
  digitalWrite(LED_BUILTIN, LOW);
  delay(100);
 }
}
```

Bibliografía

Arduino. (s.f.). *Arduino: Tecnología para todos*. Obtenido de https://arduinodhtics.weebly.com/

Arduino, P. c. (2018). *Sensores de Flujo de Agua para Arduino.* Obtenido de https://proyectosconarduino.com/sensores/flujo-caudalimetro/

Arduino, P. c. (2018). *Sensores de Polvo y Partículas en el Aire.* Obtenido de https://proyectosconarduino.com/sensores/polvo-particulas/

Arduino, P. c. (2022). *Placa Arduino Uno R3.* Obtenido de https://proyectosconarduino.com/placas/arduino-uno-r3/

CEAC. (1 de agosto de 2018). *Tipos de placas para programar con Arduino.* Obtenido de https://www.ceac.es/blog/tipos-de-placas-para-programar-con-arduino

Crespo, E. (2017). *Aprendiendo a manejar Arduino en profundidad.* Obtenido de https://aprendiendoarduino.wordpress.com/2017/06/18/ide-arduino-y-configuracion/

Dieguez, L. (2 de julio de 2020). *Sensores mas usados con Arduino*. Obtenido de https://kolwidi.com/blogs/blog-kolwidi/sensores-mas-utilizados-con-arduino

E., V. (9 de enero de 2021). *Arduino, ¿qué placa escojo?, ¿por dónde empiezo?* Obtenido de https://draco-robotic.com/arduino-que-placa-escojo-por-donde-empiezo/

Eectronic, U. (https://uelectronics.com/producto/modulo-ky-037-sensor-de-sonido/). *Módulo KY-037 Sensor de Sonido.* Obtenido de https://uelectronics.com/producto/modulo-ky-037-sensor-de-sonido/

Elecronic, I. (5 de Marzo de 2021). *¿Qué es un Arduino y para qué sirve?* Obtenido de https://inelectronic.com/que-es-un-arduino-y-para-que-sirve/

Electronics, M. (s.f.). *Como usar y administrar librerías*. Obtenido de https://arduino.cl/como-usar-y-administrar-librerias/

Electronics, U. (s.f.). *Sensor Receptor Infrarrojo IR Módulo KY-022.* Obtenido de https://uelectronics.com/producto/sensor-receptor-infrarrojo-ir-modulo-ky-022

Electronics, U. (s.f.). *Sensor Vibración Módulo KY-002.* Obtenido de https://uelectronics.com/producto/modulo-ky-002-sensor-vibracion

Fernández, Y. (23 de septiembre de 2022). *¿Qué es Arduino, cómo funciona y qué puedes hacer con uno?* Obtenido de xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

Fundación, A. (s.f.). *¿Sabes qué es un Arduino y para qué sirve?* Obtenido de https://www.fundacionaquae.org/wiki/sabes-arduino-sirve/

Hernández, L. d. (s.f.). *Curso de Arduino aprende a programar desde cero.* Obtenido de https://programarfacil.com/blog/arduino-blog/curso-de-arduino/

INDUSTRUINO, P. (s.f.). *Historia de Arduino.* Obtenido de https://www.bolanosdj.com.ar/MOVIL/ARDUINO2/HistoriadeArduino.pdf

Lamas, L. (2018). *ESQUEMA DE PATILLAJE (PINOUT) DE ARDUINO UNO, NANO, MINI Y MEGA.* Obtenido de https://www.luisllamas.es/esquema-de-patillaje-de-arduino-pinout/

Lamas, L. (9 de enero de 2018|). *MEDIR DISTANCIA CON PRECISIÓN CON ARDUINO Y SENSOR LÁSER VL53L0X Y VL6180X.* Obtenido de https://www.google.com/search?q=Medir+distancia+con+precisi%C3%B3n+con+Arduino+y+sensor+l%C3%A1ser+VL53L0X+y+VL6180X+se+puede+lograr.+El+VL53L0X+es+un+sensor+de+distancia+infrarrojo+l%C3%A1ser+de+%C3%BAltima+generaci%C3%B3n%2C+que+podemos+emplear+junto+co

Llamas, L. (16 de marzo de 2015). *MEDIR NIVEL DE LUZ CON ARDUINO Y FOTORESISTENCIA LDR (GL55).* Obtenido de https://www.luisllamas.es/medir-nivel-luz-con-arduino-y-fotoresistencia-ldr/

Llamas, L. (29 de marzo de 2016). *MEDIR TEMPERATURA Y HUMEDAD CON ARDUINO Y SENSOR DHT11-DHT22.* Obtenido de https://www.google.com/search?q=Medir+temperatura+y+humedad+con+Arduino+y+sensores+DHT11+y+el+DHT22+(o+AM2302)+son+dos+modelos+de+una+misma+familia+de+sensores%2C+que+permiten+realizar+la+medici%C3%B3n+simult%C3%A1nea+de+temperatura+y+humedad.+Estos+senso

Llamas, L. (enero de 2018). *DETECTAR GESTOS CON ARDUINO Y SENSOR APDS-9960.* Obtenido de https://www.luisllamas.es/detectar-gestos-con-arduino-y-sensor-apds-9960/

Llamas, L. (2018). *DETECTOR DE LLUVIA CON ARDUINO Y SENSOR FC-37 O YL-83.* Obtenido de https://www.luisllamas.es/arduino-lluvia/

maker, D. M. (s.f.). *Esquema completo sobre el Arduino Uno*. Obtenido de https://dorcu.com/esquema-completo-sobre-el-arduino-uno/

Mastoner. (octubre de 2022). *Blog Mas Toner.* Obtenido de https://mastoner.com/blog/que-es-arduino-y-para-que-sirve/

Mechatronic, N. (s.f.). *TUTORIAL DE ARDUINO Y SENSOR ULTRASÓNICO HC-SR04*. Obtenido de https://naylampmechatronics.com/blog/10_tutorial-de-arduino-y-sensor-ultrasonico-hc-sr04.html

Programarfácil. (2018). *Arduino UNO R3 la revolución del hardware libre.* Obtenido de https://programarfacil.com/blog/arduino-blog/arduino-uno-r3/

Prometec. (s.f.). *LOS SENSORES INFRARROJOS.* Obtenido de https://www.prometec.net/siguelineas-ir/

Sixto, J. D. (s.f.). *Sistema Controlado por sustema Arduino Uno*. Obtenido de https://www.studocu.com/es-mx/document/universidad-estatal-de-sonora/mec11c1/sistema-controlado-por-la-tarjeta-arduino-uno/11811582