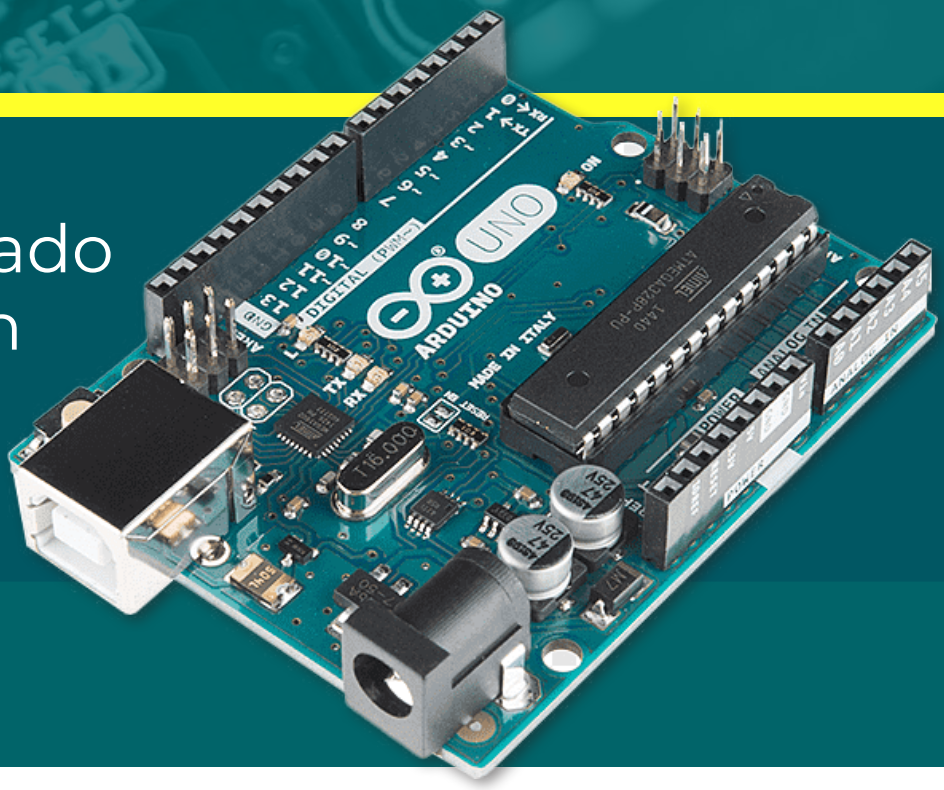
Universidad Central

Facultad de Ingeniería y Arquitectura

Aprendizaje basado en proyectos con **ARDUINO**



2022

Autores
Nelson Sepúlveda Navarro
Yonnhatan García Cartagena

< FINARQ UCEN >

ISBN 978-956-330-075-8
Sello Editorial Universidad Central: 978-956-330

Nelson Sepúlveda, Doctor en Ciencias c/m en Física, profesor de Física y Matemáticas, es académico del departamento de Ciencias Básicas, de la Escuela de Ingeniería de la **Universidad Central de Chile**. Ha generado una línea de trabajo en el área STEAM con Arduinos desde la Vinculación con el Medio.

Yonnhatan García, Profesor de Física y Ciencias Naturales, es académico del Departamento de Física de la **Universidad Metropolitana de Ciencias de la Educación**. Desde esa institucion impulsó el desarrollo del Laboratorio de Tecnologia, **LabTec-UMCE**. Actualmente, el LabTec es promotor activo de la educación STEM y STEAM en vínculo con otras organizaciones.

INDICE

Presentación

El siguiente texto es un conjunto de ideas y propuestas para el uso de la metodología de Aprendizaje Basado en Proyectos (ABP), por medio de la incorporación de ciencia y tecnología con el uso del interfaz Arduino UNO. Las propuestas de actividades que encontrará en este documento han sido contextualizadas al currículum escolar chileno, con la intención de incorporar el ABP en cursos de ciencias y, en particular, en Ciencias para la Ciudadanía. El trabajo en base al ABP permite abordar proyectos semestrales o anuales, incorporando las preguntas: ¿Cómo mido lo que observo? ¿Cómo interpreto lo que mido? ¿Cómo funcionan los instrumentos de medición? ¿Cómo puedo automatizar un sistema? ¿De qué manera este proyecto soluciona un problema real de mi entorno?

En este texto se propone la realización de proyectos con componentes electrónicos de bajo costo. También incluye la simulación de circuitos por medio del software Tinkercad, disponible gratuitamente en la web.

El texto se divide en dos grandes secciones: i) Talleres introductorios, y ii) Proyectos. En los Talleres introductorios se desarrollan las nociones básicas para el trabajo con diversos componentes, los que son considerados, cada uno por separado, como bloques estructurales para el desarrollo de los proyectos. En la segunda parte, sección de Proyectos, se propone construir aplicaciones para el mundo real, donde la intención es que al desarrollar esos proyectos se pueda efectivamente contextualizar la enseñanza de las disciplinas STEM.

En las distintas actividades propuestas se darán algunos ejemplos modelados con software disponibles en la web, como Tinkercad[1] y Fritzing[2]. Tinkercad es una potente herramienta web para la simulación y modelamiento de circuitos electrónicos que pueden usarse con Arduino. Al ser virtual, elimina la barrera de costo de acceso al hardware o componentes electrónicos. No obstante, es un servicio web que requiere de conexión a internet. Por su parte, el software Fritzing dispone de una importante cantidad de componentes electrónicos para el armado de circuitos en un protoboard. También ayuda a generar los diagramas esquemáticos y placas electrónicas para los circuitos que han sido creados. Si bien trabaja directamente en la computadora, sin requerir conexión a internet, no permite la simulación ni programación de los circuitos basados en Arduino.

Este texto pretende contribuir tanto al desarrollo de actividades para la enseñanza media como en la realización de cursos introductorios, complementarios o de extensión en la enseñanza superior. Con todo, la incorporación de la metodología ABP en los cursos de ciencias permite trabajar en proyectos transdiciplinarios, contextualizados a realidades territoriales particulares.

A partir de las actividades aquí presentadas, que involucran ciencia, matemática, electrónica, computación y comunicación, se puede promover el cuidado del medio ambiente cumpliendo con Objetivos de Desarrollo Sostenible, así como metodologías activas de aprendizaje y, en especial, la alfabetización digital, estimulando la curiosidad por los fenómenos científicos.

Esta publicación surge como una colaboración entre la Escuela de Ingeniería de la Universidad Central de Chile y el Laboratorio de Tecnología LabTec[3] de la Universidad Metropolitana de Ciencias de la Educación. La intención es facilitar el acceso a las tecnologías emergentes en el campo educacional y fomentar su uso en la docencia, impulsando así la innovación mediante la creación de tecnología que dé soporte a los procesos de enseñanza en el área de las ciencias y de la tecnología.

---

1 *https://www.tinkercad.com/*
2 *https://fritzing.org/*
3 *http://bit.ly/labtec-UMCE*

Método de Trabajo

La Unidad de Currículum y Evaluación (UCE) del Ministerio de Educación en Chile publicó en 2019 el texto Metodología de Aprendizaje Basado en Proyectos[4] para el nuevo currículo de 3° y 4° medio. Este documento destaca como eje de acción el desarrollo de habilidades para el siglo XXI; en especial, pensar con creatividad e innovación, el trabajo colaborativo, el pensamiento crítico y la comunicación.

En el caso de Ciencias para la Ciudadanía, en dicha publicación el MINEDUC sostiene que la metodología STEM[5] permite al estudiante aprender que las matemáticas y las ciencias, junto con la tecnología, son herramientas necesarias para ayudar a identificar problemas, recopilar y analizar datos, modelar fenómenos, probar las posibles soluciones y resolver los problemas, tanto los que se presentan en la vida profesional como en la vida diaria. Los saberes en los cuales se enfoca el currículo de Ciencias para la Ciudadanía[6], se estructuran en cuatro grandes ejes temáticos: Bienestar y Salud; Ambiente y Sostenibilidad; Seguridad, prevención y autocuidado; y Tecnología y Sociedad.

El Aprendizaje Basado en Proyectos (ABP) puede aportar mucho al desarrollo de las habilidades para el siglo XXI. También se puede utilizar como estrategia didáctica en los cursos de "Ciencias para la Ciudadanía" o "Ciencia, Tecnología y Sociedad", para desde allí trabajar en el logro de los Objetivos para Desarrollo Sostenible (ODS).

La propuesta de este texto es que dichos objetivos pueden ser trabajados junto a la alfabetización digital a partir del uso de Arduino, tecnología que es muy popular por su facilidad de uso, por su bajo costo y por ser de código abierto. De esta forma, se puede proveer de un ambiente altamente contextualizado con aplicaciones al mundo real.

El ABP forma parte de lo que se denomina "aprendizaje activo" y posee los principios básicos del modelo constructivista de aprendizaje. En contraposición con las metodologías tradicionales, en el ABP el o la docente, si bien juegan un rol fundamental, no son los protagonistas del proceso de aprendizaje de sus estudiantes, si no que toman un rol de guía, mediador y/o facilitador entre estos y el conocimiento.

El ABP se caracteriza por exigir a los y las docentes una constante actualización de sus propios conocimientos, debido a que las temáticas y metodologías de trabajo e investigación que escogen sus estudiantes no siempre son de su total dominio, lo que los lleva a ampliar sus conocimientos, a través del diálogo con sus colegas, y a realizar sus propias investigaciones, con la finalidad de responder a las preguntas de sus alumnos y alumnas a través de su propia experiencia. Esto conlleva que el profesorado debe ser capaz de organizar y animar situaciones de aprendizaje, utilizar las nuevas tecnologías, implicar a los estudiantes en su aprendizaje y en su trabajo y gestionar la progresión de los aprendizajes.

Por su parte, el rol del estudiante dista mucho de la forma de trabajo tradicional en aula, donde es un mero receptor pasivo de la información, pasando a ser agentes activos y protagonistas en su proceso de aprendizaje a través de la interacción con el mundo real.

---

4 https://www.curriculumnacional.cl/portal/Curso/Tecnico-Profesional/3-Medio-TP/140166:Metodologia-de-aprendizaje-basado-en-proyectos

5 STEM: Sigla en inglés surgida de la agrupación de ciencia, tecnología, ingeniería y matemáticas.

6 https://www.curriculumnacional.cl/portal/Formacion-General/Plan-Comun-de-formacion-general/Ciencias-para-la-ciudadania/

En general, el ABP se puede caracterizar a partir de las siguientes etapas:

1. Definición del problema: En esta etapa, el profesor presenta la problemática o pregunta desafiante en torno a la cual se trabajará. Aquí se debe indagar acerca de cuáles son los conocimientos previos relacionados con la temática central. Es importante que el problema sea presentado de manera que logre motivar a los y las estudiantes, ya sea destacando la importancia en su entorno, compartiendo experiencias profesionales y formulando distintas preguntas, de manera que sean ellos mismos los que lleguen a la esencia del problema.

2. Formulación del objetivo o meta: Esta etapa tiene como fin que el curso proponga el "producto" que se desee desarrollar para dar solución a la problemática, ya sea una presentación, un modelo, un experimento, un aparato tecnológico, etc. Es importante que el docente vele por la viabilidad del proyecto, considerando los recursos materiales y humanos que se requieren y el tiempo que se dispone en el global de la unidad.

3. Formación de equipos: En este momento se conforman los distintos grupos de trabajo y se asignan roles y tareas entre los integrantes. Es importante que las y los estudiantes tengan autonomía en la organización de tareas dentro del equipo, pero siempre debe existir la orientación del docente para sugerir dinámicas internas dentro de este y apoyar en la generación de ideas para encauzar el desarrollo del proyecto. En el ABP es deseable que los grupos de trabajo sean heterogéneos y que en todo momento se vele por un trabajo colaborativo equilibrado entre los mismos estudiantes y otros miembros de la comunidad.

4. Investigación: En este paso, el grupo de trabajo investiga acerca de la temática central, recopilando información y datos de fuentes confiables que pueden ser recomendadas por el o la docente.

5. Elaboración del proyecto: En esta etapa, el grupo establece los pequeños objetivos que se deben conseguir para alcanzar la meta final o producto. Se deben tomar decisiones, resolver problemas, considerar los recursos y el tiempo destinado para cada tarea. De acuerdo con la dificultad del proyecto, los estudiantes necesitarán más o menos orientación en su trabajo. Es fundamental que el docente esté atento tanto a las dificultades como a las oportunidades de aprendizaje que vayan surgiendo a lo largo del proceso.

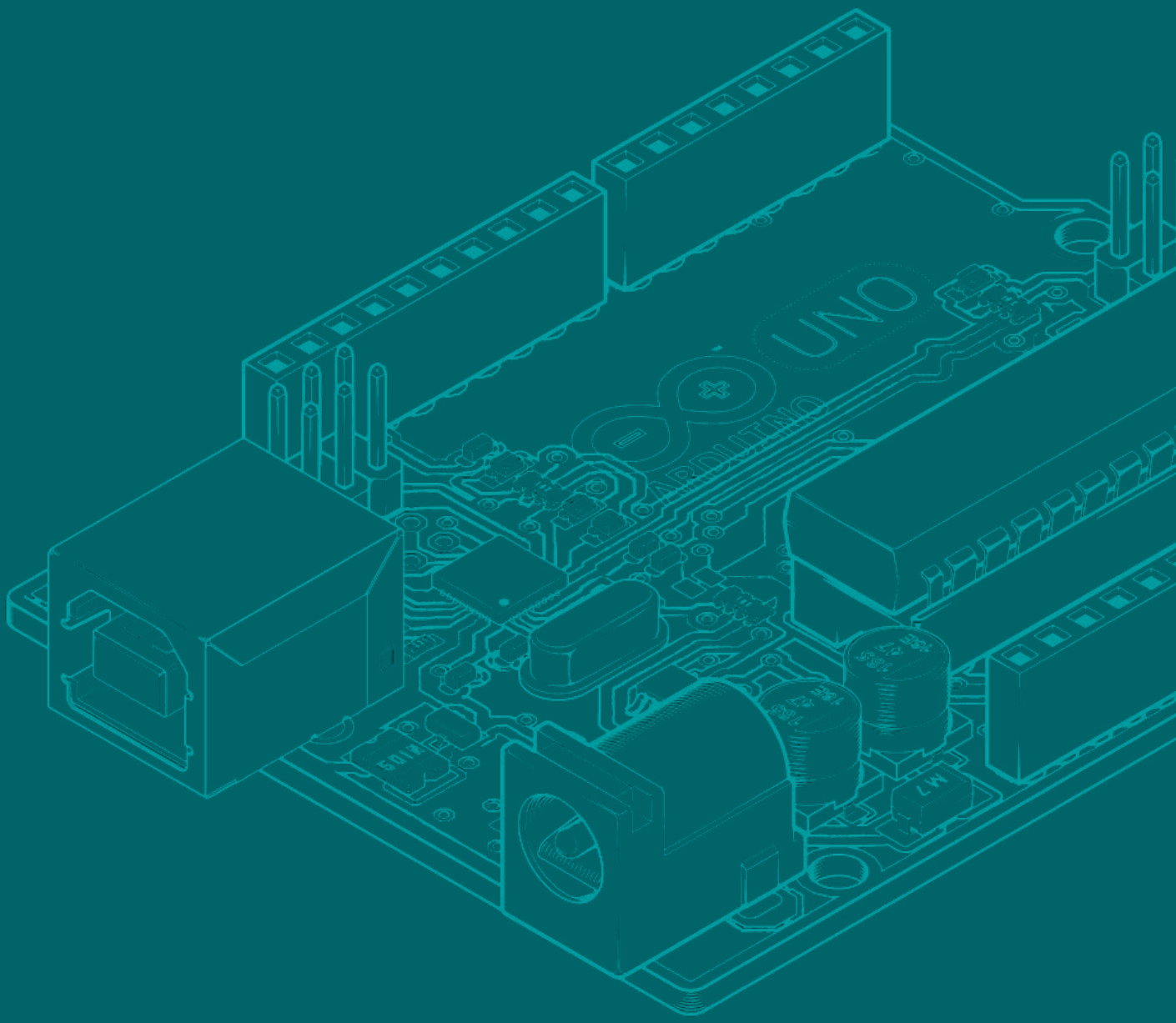
6. Comunicación: En esta última etapa, se presenta el proyecto a los pares y a los otros miembros de la comunidad. Aquí se requiere de una preparación previa por parte de los estudiantes y lo deseable es que se apoyen de herramientas como las TIC, de ser posible, para difundir los resultados del proyecto por diversos medios.

7. Evaluación: Finalmente, se revisan los procesos, se plantean mejoras, se analiza el desempeño y se adquieren compromisos de perfeccionamiento. En esta última etapa se evalúan todos los procesos que llevaron a conseguir el producto final. Es deseable que se incluya una autoevaluación y una coevaluación, en donde se consideren aspectos relacionados con la motivación que experimentaron las y los estudiantes a la hora de afrontar las distintas etapas. Se sugiere trabajar con un portafolio, diario de aprendizaje y/o rúbricas.

A partir de este momento, dejamos en sus manos este texto que esperamos sirva para promover el ABP y el desarrollo de habilidades para el siglo XXI desde las aulas.

SECCION I

TALLERES
INTRODUCTORIOS



TALLER 1

{ Introducción a ARDUINO

Una placa Arduino es un dispositivo electrónico compuesto por un microcontrolador llamado ATMEGA328, que le permite procesar datos y realizar diversas tareas como: mover un motor, detectar la presencia de un objeto, medir el tiempo entre dos eventos, encender o apagar una luz, etc. La placa Arduino UNO puede ser programada por medio de un software open source o software libre, es decir, un software que no necesita una licencia para ser instalado. Este software se puede descargar desde la página oficial de la fundación Arduino[7].

La placa Arduino posee diversos pines donde se conectan cables (ver figura 1.1). Esto permite controlar sensores, motores o pantallas LCD sin tener que soldar dichos cables.

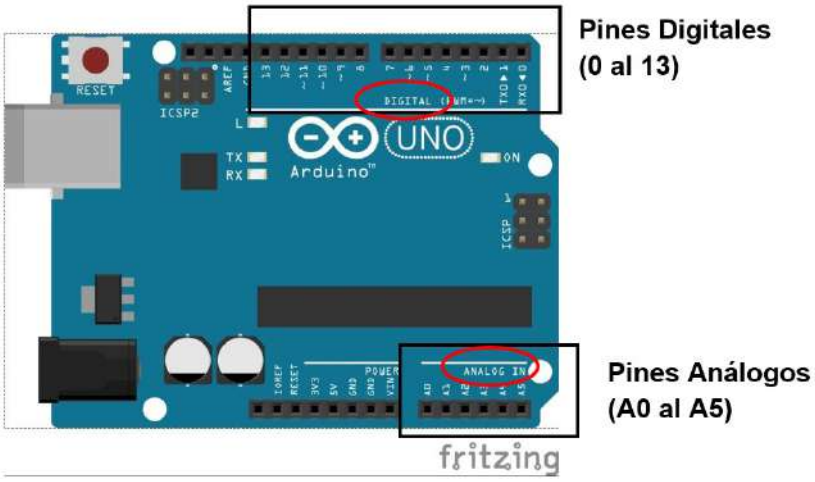


Figura 1.1: Arduino UNO esquematizado por medio del software Fritzing.

En lo pines mostrados en la imagen 1.1 se pueden incorporar sensores para monitorear múltiples parámetros del mundo físico. Para ello se debe programar el Arduino de manera que pueda "saber" que tiene conectado un sensor determinado a un pin (digital o análogo) y especificar cuál pin recibe la señal o los datos. Por otra parte, si por ejemplo se debe conectar una pantalla LCD, hay que programar el Arduino de igual manera que con los sensores, con tal de especificar que tiene conectada una pantalla y que muestre un texto o datos.

En resumen, Arduino es una interfaz capaz de conectar el mundo digital con el mundo físico y el Arduino UNO puede ser programado en sus pines como entradas (sensores o botones) o salidas (pantallas o alertas de luces o sonido).

También es posible que los datos obtenidos de los sensores puedan ser enviados a dispositivos como computadores o dispositivos móviles. Por ello, el Arduino resulta eficiente para ser utilizado como sistema de alerta o de monitoreo. En la figura 1.1 se observan dos tipos de pines: estos están clasificados en dos grupos, los cuales se pueden programar como salidas o entradas de tipo analógicas o digitales.

El Arduino Uno posee seis pines analógicos, desde el A0 al A5, los cuales son capaces de leer valores de voltaje de 0 a 5 Volts con resolución de 10 bits. Al considerar el valor máximo de entrada 5V dividido por la resolución, se obtiene que es capaz de detectar cambios de 0.004882814V (5/1024 Volts). Esto significa que podríamos detectar cambios en señales de entrada desde 5mV aproximadamente. Un sensor de temperatura, de pH o de UV, no mide el parámetro de manera directa, sino que registra cambios de voltajes o de resistencias del sensor mismo. Luego, por medio de una función o calibración podemos interpretar ese cambio de voltaje "0.004882814V" como un cambio en el parámetro que estamos midiendo. En otras palabras, un sensor es un dispositivo que convierte una señal del mundo físico (temperatura, presión, humedad, radiación UV, etc.) en una señal eléctrica que puede ser recogida e interpretada mediante la placa microcontrolada Arduino.

7 https://www.arduino.cc

Por otra parte, son 14 los pines digitales del Arduino Uno, los que van desde el pin 0 al pin 13. A diferencia de los pines analógicos, los pines digitales pueden tener solo dos valores, alto o bajo, interpretables como uno o cero, los que en la realidad corresponden a valores de 0 o 5 Volts.

El trabajo con Arduino puede realizarse de forma real, con componentes electrónicos tangibles (hardware) o valiéndose de simulaciones mediante el uso de software. Cuando no se cuente con el hardware necesario, se pueden realizar cada una de las actividades de los talleres de este texto por medio del software online Tinkercad. Para trabajar con este programa no es necesario descargar o instarlo, solo se debe crear una cuenta para trabajar online.

AUTODESK® TINKERCAD®

Para crear una cuenta de Tinkercad®, primero se debe ingresar a la página web http://www.tinkercad.com y unirse como se muestra en la figura 1.2. Posteriormente confirmar si ingresará a una clase como profesor o estudiante, o si lo hará de forma privada con una cuenta de correo, cuenta Google o Apple.

Figura 1.2: Página home de Tinkercad®.

Luego se debe ir al ícono Circuitos, en la parte izquierda bajo el nombre de ingreso, y ya estamos listos para crear nuestro primer circuito desde el ícono verde "Crear un nuevo Circuito" (ver figura 1.3).

Figura 1.3: Ingreso a creación de circuitos en Tinkercad®.

Una vez en el circuito nuevo, desde la pestaña "Componentes" es posible obtener componentes electrónicos básicos o componentes específicos de Arduino, como por ejemplo el Arduino UNO (ver figura 1.4). Para trabajar con cada componente, lo siguiente consiste en seleccionar y arrastrar.

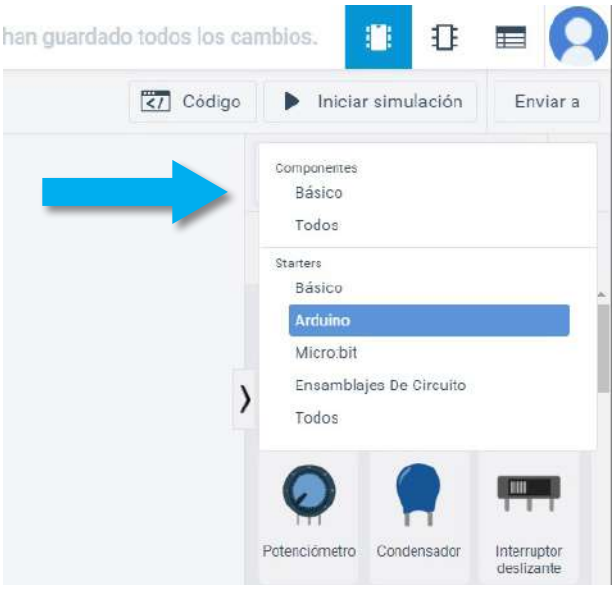


Figura 1.4: Ventana principal de Tinkercad® para simulación de circuitos.

Desde la pestaña de "componentes" es posible obtener los componentes y sensores, como el Arduino UNO, con tal de construir los circuitos e iniciar una simulación. Las funciones esenciales de cualquier programa en Arduino se resumen a continuación.

Funciones esenciales en un programa de Arduino

Sea que usted programe Arduino mediante Tinkercad, o valiéndose de la interfaz IDE instalada en su computadora, cualquier programa que cree utilizará las siguientes funciones esenciales.

void setud ( ) { }

En esta parte se escriben las instrucciones que es necesario definir solo una vez. Todo lo que esté entre los corchetes se realizará solo una vez cuando el Arduino Uno esté funcionando.

void loop ( ) { }

Está función permite que todo lo que esté entre corchetes se repita indefinidamente mientras el Arduino esté funcionando. El Arduino se alimenta directamente desde el puerto USB del computador. También se puede alimentar con un transformador o con una batería externa como las que se utilizan para el celular, lo que permite darle autonomía al Arduino.

En el caso de conectar un sensor en alguno de los pines de Arduino, primero será necesario "leer" el pin correspondiente. Para ello se debe escribir una variable, por ejemplo "lectura", e indicar el pin desde donde leemos la entrada, que puede ser análoga o digital. En el caso de ser análoga, serán utilizados los pines definidos como A0 hasta A5. En cada caso, la instrucción para leer la entrada análoga será:

lectura=analogRead(pin);

La función de entrada escrita de esta manera entregará un valor que va desde 0 a 1023, proporcional al nivel de la señal de entrada.

Por otra parte, las entradas digitales están definidas en los mismos pines donde se encuentran las salidas digitales. En este caso, se hace referencia a los pines que van del 0 hasta el 13. A diferencia de las entradas analógicas, estas son capaces de "entender" solo dos niveles de señal: LOW, que corresponde al valor 0V, y HIGH, que corresponde al valor de 5V. Puede parecer una desventaja, pero en realidad no es así. No solo porque, a veces, únicamente necesitemos saber dos estados (interruptor, pulsador, sensor de presencia, etc.), sino porque así es capaz de leer señales de pulsos digitales que cambian sistemáticamente entre valores bajos (LOW) y altos (HIGH).

Por ejemplo, un sensor analógico de temperatura como es el LM35 incrementaría el valor del voltaje que llega a la placa de forma proporcional a la temperatura. Sin embargo, un sensor digital como el ds18b20 lo que haría es cambiar la sucesión de pulsos y, por tanto, el mensaje que contiene el valor de la temperatura. Si bien por defecto los pines digitales vienen configurados como entrada, también es posible declararlos manualmente como entrada mediante la expresión siguiente:

pinMode(pin,INPUT);

Donde, pinMode es el comando que define al pin digital, pin es el número asociado entre 0 y 13, e INPUT corresponde a lectura. Para almacenar los valores posibles, LOW o HIGH, podemos definir la variable lectura.

lectura=digitalRead(pin);

Arduino UNO posee tres tierras (GND) y dos salidas de voltaje que le servirán para alimentar sensores o pantallas desde el Arduino (ver imagen 1.5). Por último, es muy bueno saber que todo lo que sigue luego de // es un comentario. Esto es muy útil para que declare todo lo que está haciendo y no olvide para qué sirve cada comando, al igual que al comenzar un programa: si este es de su autoría, escriba su nombre, fecha de creación, versión del programa que está realizando, etc., lo cual será útil cuando realice modificaciones y actualizaciones.

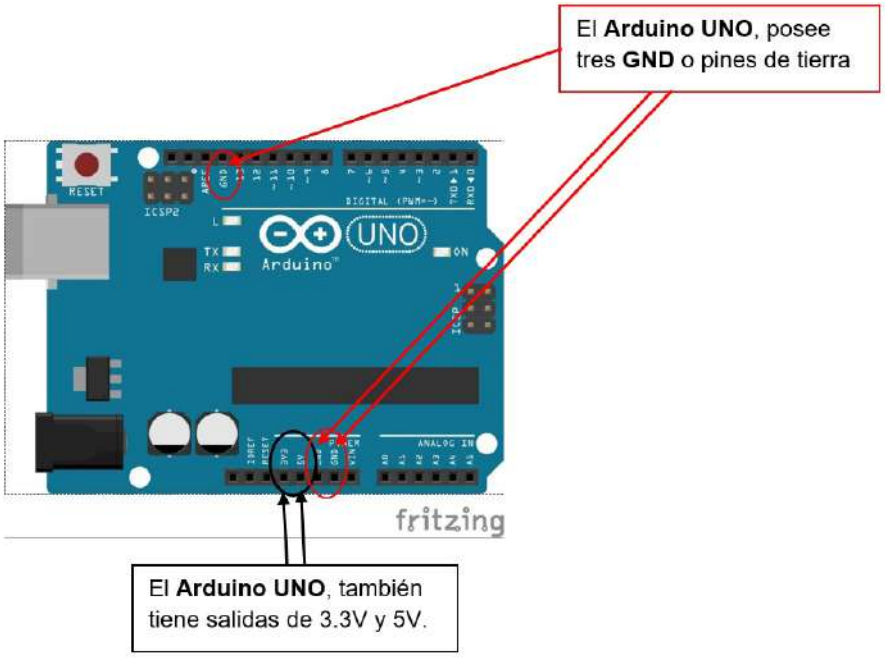


Figura 1.5: Tierras y salidas para alimentar led, sensores o pantalla LCD.

En síntesis, Arduino UNO posee:

- Puertos digitales: pines 0-7 y pines 8-13.
- Puerto analógico: pines A0-A5.
- Puerto de energía: 5V de salida y 3.3V de salida, GND o tierra del circuito.

Los códigos esenciales revisados anteriormente, se deben cargar en el Arduino a través del entorno de programación IDE (ver figura 1.6), el cual permite que la programación realizada en el computador, se integre a la placa arduino mediante el cable USB.

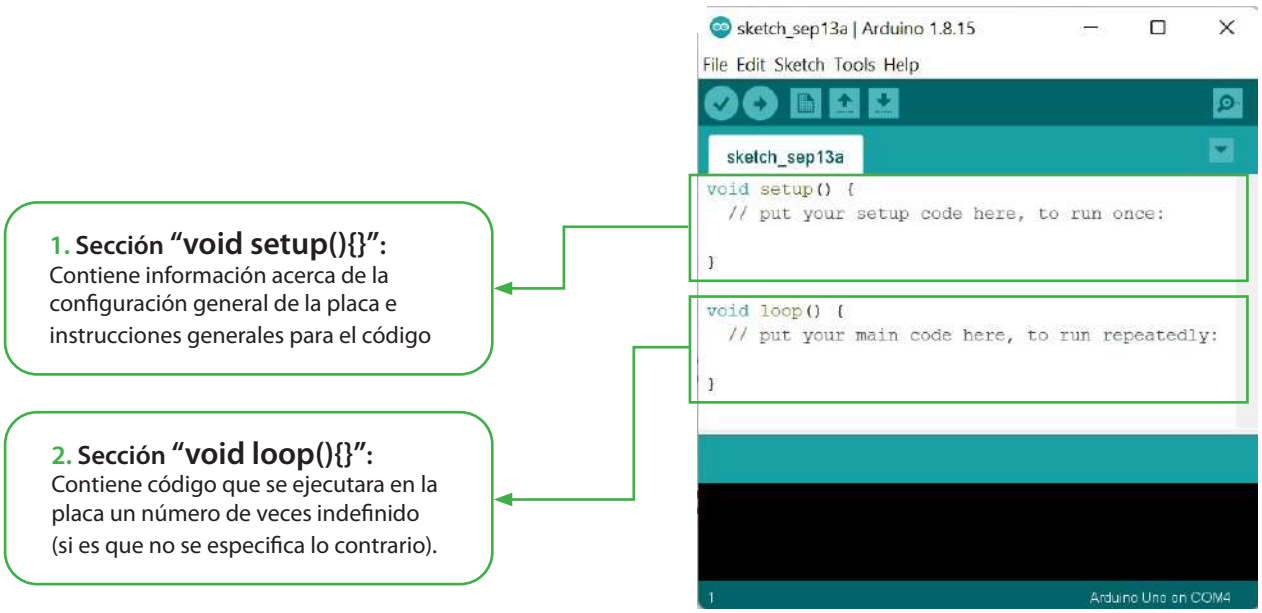


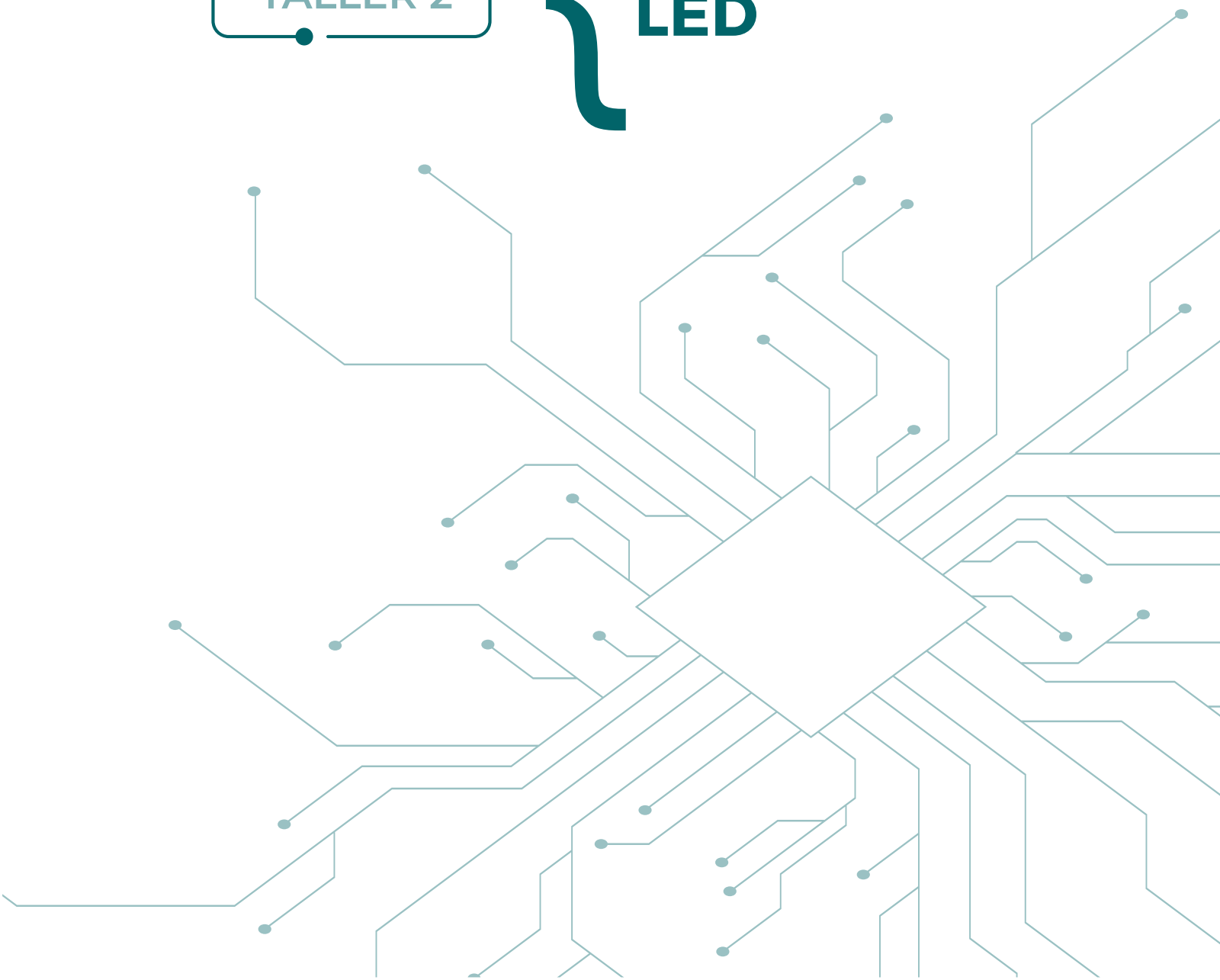
Figura 1.6: Entorno de programación IDE de Arduino.

Desafío

Conecte la placa Arduino a la computadora y explore la interfaz de programación IDE. Si no posee una placa Arduino real, explore el simulador Tinkercad disponible en la web https://www.tinkercad.com.

TALLER 2

{ Control de un **LED**



Ahora que ya se conocen algunos elementos esenciales de la placa Arduino, y ya instalado el software necesario, se está en condiciones de realizar el primer programa, consistente en el encendido y apagado de led con cierta frecuencia controlada por el programador.

Este taller consiste básicamente en poner en funcionamiento la placa Arduino UNO, para lo cual se trabajará con las librerías de ejemplos que posee el software IDE de Arduino. El objetivo particular es tomar el control de un led por medio de las salidas digitales. Esto significa no solo alimentarlo y que este encienda, sino también controlar los tiempos cuando estará encendido y cuando está apagado, es decir, consiste en programar el parpadeo del led.

Componentes

- protoboard
- Arduino UNO
- Cables varios
- 1 Resistencia eléctrica 330Ω
- 1 led

Opcionalmente, se puede usar el modelo disponible en Tinkercad:

- Parpadeo: https://www.tinkercad.com/things/15sgLumXoiU
- Comparación de circuitos con un led: https://www.tinkercad.com/things/6FBCCZejpNa

Desarrollo

Las conexiones eléctricas se realizan por medio de cables en un protoboard. Este último corresponde a una placa de prueba que permite agregar o quitar cables, armar circuitos prototipos, sin necesidad de soldar los sensores o los cables. Las conexiones internas de un protoboard están unidas en los extremos a lo largo de filas y, en el centro, en columnas de 5 pines conectados entre ellos (ver figura 2.1).

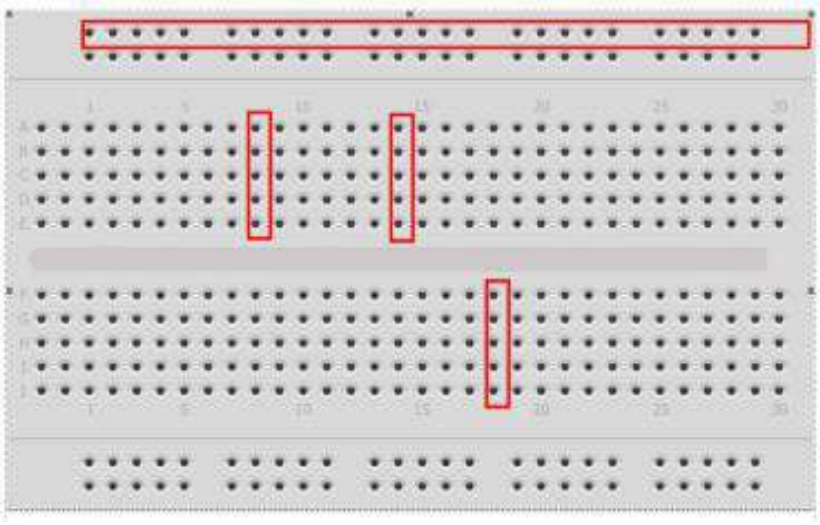


Figura 2.1. Protoboard indicando fila superior interconectada, y columnas también interconectadas según se indica en los rectángulos rojos.

Las resistencias eléctricas se caracterizan por ser un dispositivo cuya función es proporcionar más o menos portadores de carga a un circuito. Su valor puede ser medido directamente por un óhmetro o multímetro, y también puede ser leído por medio de un código de colores. Cada resistencia posee cuatro franjas de colores y cada color representa un número, como muestra la tabla 2.1.

Tabla 2.1. Código de colores resistencias eléctricas

Negro	Café	Rojo	Naranjo	Amarillo	Verde	Azul	Violeta	Gris	Blanco	Oro	Plata
0	1	2	3	4	5	6	7	8	9	5%	10%

Las dos primeras franjas representan un número de decena y la unidad, respectivamente; la tercera franja es un multiplicador que corresponde al número, que va de exponente en base diez; y la cuarta franja, llamada tolerancia, generalmente es plateada o dorada y corresponde al 10% o 5%, respectivamente, de tolerancia del valor indicado, como muestra la figura 2.2.

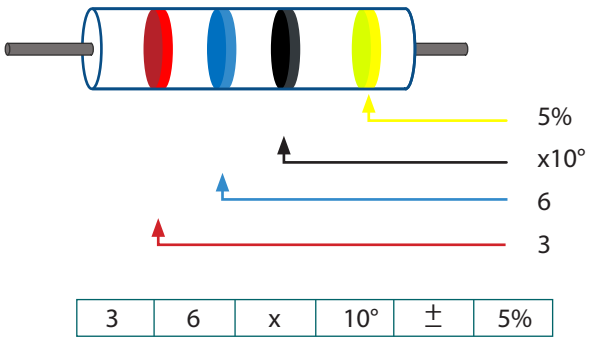


Figura 2.2. Esquema código de colores para resistencia de 36Ω.

En la figura anterior, se observa una resistencia de 36 multiplicada por una potencia de diez con exponente cero. La tolerancia del 5% indica que, en la realidad, el valor de la resistencia puede variar entre 36 - 1.8 y 36 + 1.8, es decir, el valor medido de la resistencia estará, según el fabricante, entre 34.2Ω y 37.8Ω, respectivamente.

Por su parte, un led se caracteriza por poseer dos alambres que indican la polaridad donde deben ser conectados. El alambre más largo (ánodo) debe ser conectado a los 5.0V, y el alambre más corto (cátodo) a uno de los pines tierra (GND) del Arduino, pero siempre se debe conectar una resistencia de 330Ω en serie entre el led y los 5.0V o el pin GND (ver figura 2.3). Al realizar la conexión de esta manera, se podrá comprobar que el Arduino funciona (y la luz led también).

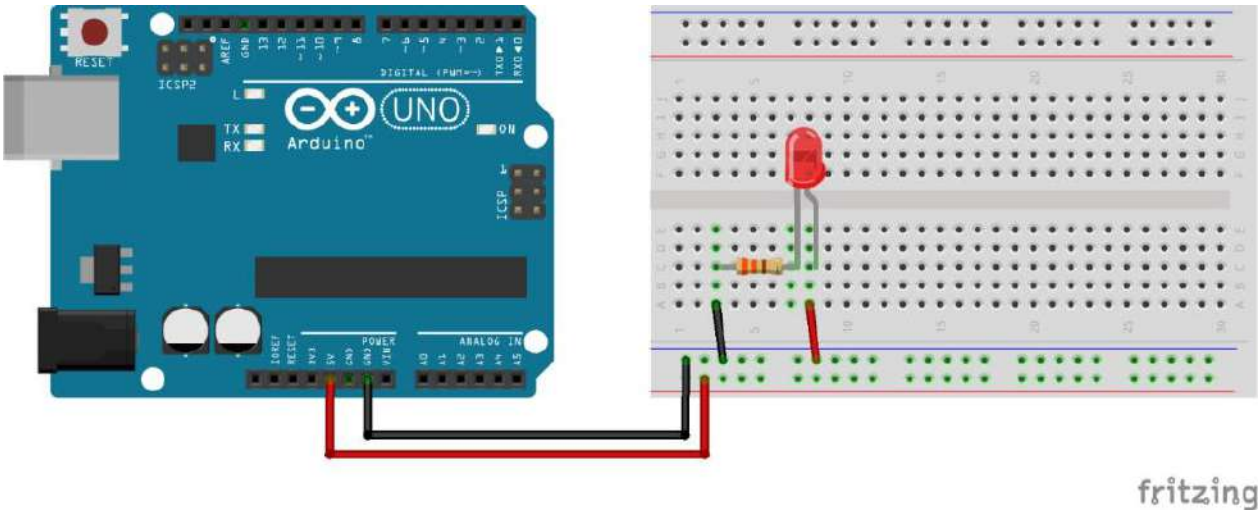


Figura 2.3. Conexión básica Arduino luz led.

Encender y apagar el led

1. Conecte en el protoboard la luz led con una resistencia de 330 Ohm en serie, el ánodo (+) del led a un pin que usted elija (la figura 2.4 muestra la conexión al pin digital 5), y el cátodo (-) a tierra del Arduino UNO (GND).

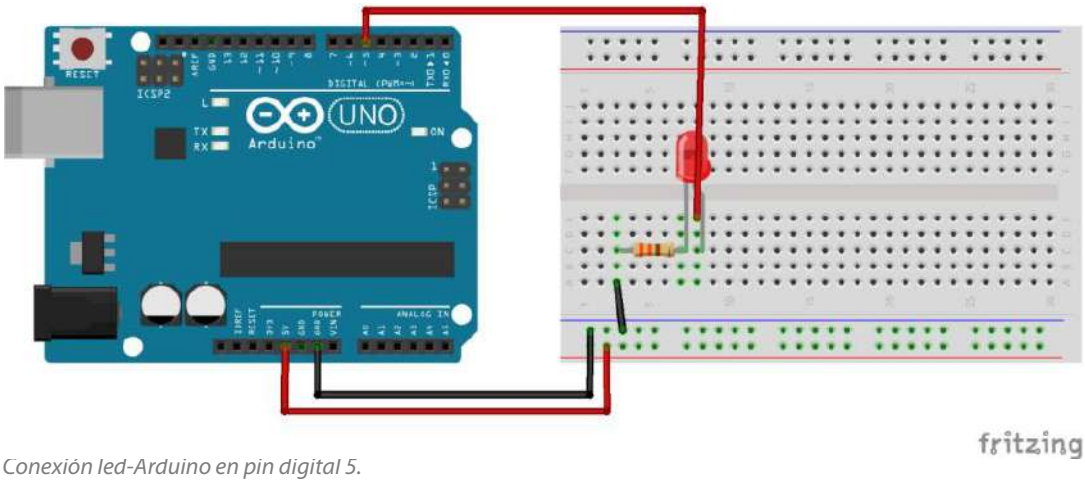


Figura 2.4. Conexión led-Arduino en pin digital 5.

2. Para el control del parpadeo de la luz led, desde *Archivo*, diríjase a la sección ejemplos del programa Arduino y cargue el ejemplo *BLINK* (parpadear), como muestra la figura 2.5.

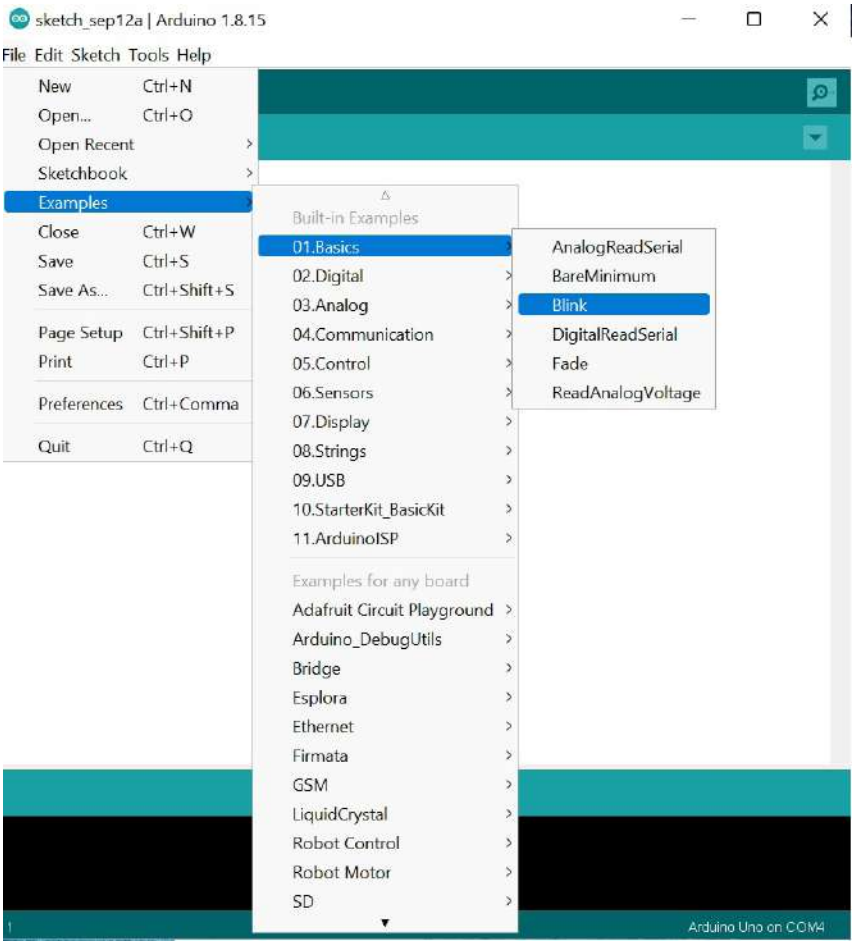


Figura 2.5. Archivo → Ejemplos → 01. Basics → Blink.

3. Debe realizar los cambios necesarios para que su led encienda y apague con el tiempo controlado, observe que en este caso solo se encenderá y apagará si se encuentra en el pin digital número 5. La placa Arduino posee un led incorporado en el pin 13, puede realizar una prueba.

Código Blink (Parpadear)

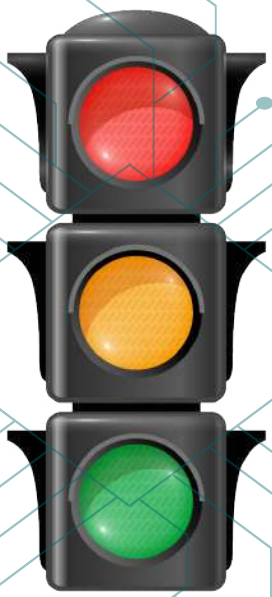
```
/*Este programa enciende un led por un segundo y luego lo apaga por un segundo, esta tarea se
repite mientras el Arduino esté encendido*/
// La función setup se repite solo una vez en este programa
void setup() {
  pinMode(5, OUTPUT);  // inicializa el pin digital 5 como una salida (output).
}
// la función loop se repetirá mientras el Arduino esté conectado a una fuente de poder.
void loop() {
  digitalWrite(5, HIGH);   // enciende el LED hacienda el voltaje HIGH
  delay(1000);                 // espera un segundo, recordar que se encuentra en milisegundos
  digitalWrite(5, LOW);    // apaga el LED hacienda el voltaje LOW
  delay(1000);                 // espera un segundo, recordar que se encuentra en milisegundos
}
```

Desafío

En el protoboard, conecte tres resistencias y tres leds de distintos colores, de tal manera de implementar su propio semáforo. ¿Todos los semáforos funcionan con los mismos tiempos de espera? ¿Los tiempos de espera son igual durante todo el día o cambian durante la tarde? Si no es así, estime los tiempos necesarios para caracterizar el funcionamiento de un semáforo.

TALLER 3

Contrucción de un **SEMAFORO** CON LED



En esta actividad se puede construir un semáforo simple con tres leds de colores. Este semáforo servirá de base para diversas otras aplicaciones donde es necesario indicar niveles o intensidades de alguna señal dada por un sensor. Por ejemplo, en el proyecto titulado "Construcción y operación de un sonómetro semáforo" se hace uso de un semáforo para indicar niveles de intensidad de sonido o nivel de ruido al interior de una habitación.

Para la realización de este taller es necesario disponer de algunos componentes electrónicos y seguir el proceso de construcción que se indica a continuación.

Componentes

- 1 protoboard
- 1 Arduino UNO
- 3 Led (rojo, amarillo, azul)
- 3 Resistencias de 220Ω
- Cables para conexiones
  Opcionalmente, se puede usar el modelo disponible en Tinkercad:
- Semáforo: https://www.tinkercad.com/things/izpu8jQ30Rq

Desarrollo

En este proyecto se construirá un semáforo con leds, como se muestra en la figura 1. Lo más importante es recordar que esta actividad consiste en disponer tres leds y programarlos para que funcionen alternadamente, de acuerdo al modelo de semáforo que comúnmente encontramos en las calles de la ciudad. Notar que este taller está estrechamente ligado al taller 2, donde se hace parpadear un led. Se sugiere armar el circuito que se muestra en la figura 3.1.

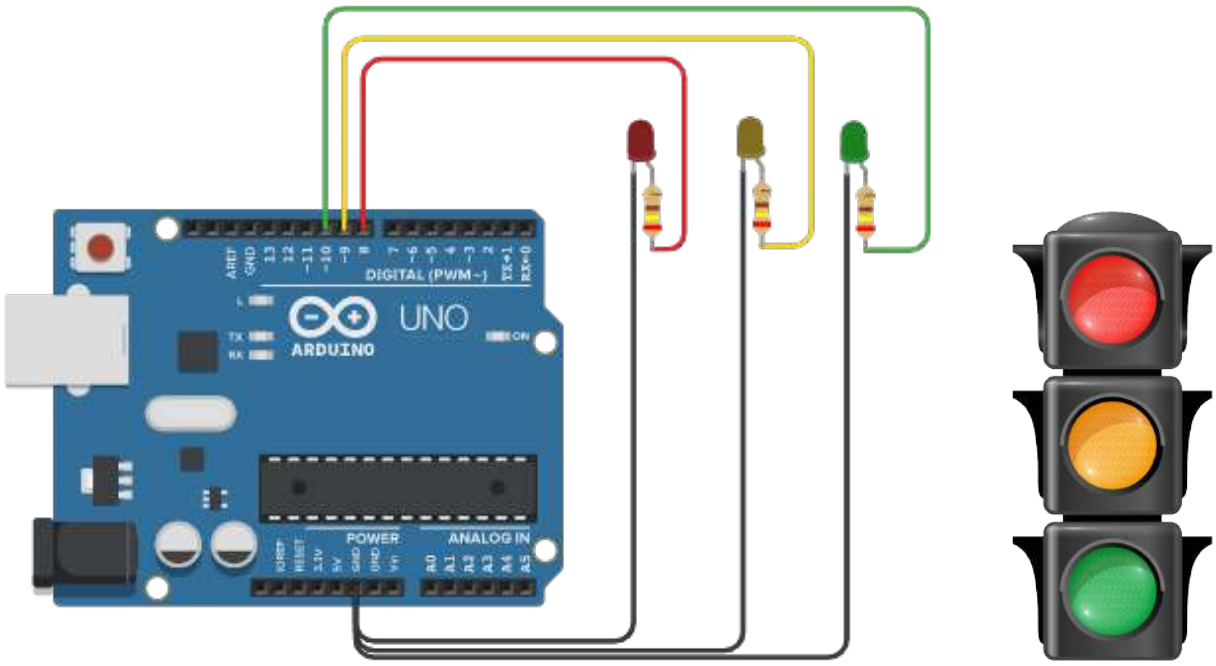


Figura 3.1. Circuito semáforo.

Código para controlar el semáforo

```
/* Ejemplo de semáforo
   by LabTec
*/

void setup() {
  pinMode(8, OUTPUT); //pin led rojo
  pinMode(9, OUTPUT); //pin amarillo
  pinMode(10, OUTPUT); //pin verde
}

void loop() {
  digitalWrite(8, HIGH);
  digitalWrite(9, LOW);
  digitalWrite(10, LOW);
  delay(1000);

  digitalWrite(8, LOW);
  digitalWrite(9, HIGH);
  digitalWrite(10, LOW);
  delay(1000);

  digitalWrite(8, LOW);
  digitalWrite(9, LOW);
  digitalWrite(10, HIGH);
  delay(1000);
 }
```

Desafíos

1. Programar el semáforo para que cada color esté encendido solo medio segundo.
2. Programar el semáforo para que el led rojo esté encendido 200 ms (milisegundos), el amarillo la mitad del tiempo del led rojo, y el verde el doble del tiempo del led rojo.

TALLER 4

Humedad y temperatura AMBIENTAL

Unos de los parámetros ambientales más importantes de registrar y controlar son la temperatura y la humedad relativa del aire. Es por ello que, en esta actividad, el objetivo consiste en poner en funcionamiento un sensor de humedad DHT11 (carcasa azul) o DHT22 (carcasa blanca), ambos mostrados en la figura 3.1. Este tipo de sensor es bastante particular, ya que es capaz de medir dos parámetros a la vez, desde una señal digital.

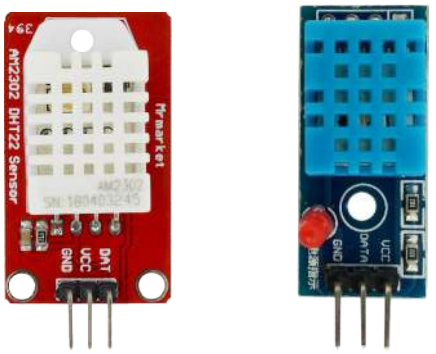


Figura 3.1. DHT22 y DHT11 indicando pines GND, 5V y Datos.

Componentes

- Protoboard
- Arduino UNO
- Cables
- Sensor de Humedad y Temperatura DHT11

Desarrollo

En los talleres anteriores se ha visto que el Arduino UNO posee **entradas y salidas digitales**, las cuales se ubican entre los pines del 0 al 13, y también admite **entradas análogas** en los pines A0 hasta el A5. En el TALLER 2 se trabajó con los pines digitales como salida, usted podía encender o apagar un led de manera controlada enviando HIGH-5V o LOW-0V. En este Taller se utiliza el sensor DHT, pues envía un tren de ondas digitales, las cuales se interpretan como temperatura y como humedad relativa. Entonces, se debe conectar nuevamente a un pin digital, pero esta vez debe leer el pin. ya que corresponde a una entrada de variables.

Para la programación de este tipo de sensores DHTXX es necesario instalar librerías desde el software IDE de Arduino. Estas librerías responden a parches para la conexión eficaz entre su computador, los sensores y el Arduino UNO. Lo primero que se instalará para la medición del sensor es la librería ***Adafruit Unified Sensor***. Una vía de instalación es desde el mismo software Arduino siguiendo la secuencia Programa → Incluir Librería → Gestionar Librerías, como se muestra en la figura 3.2.

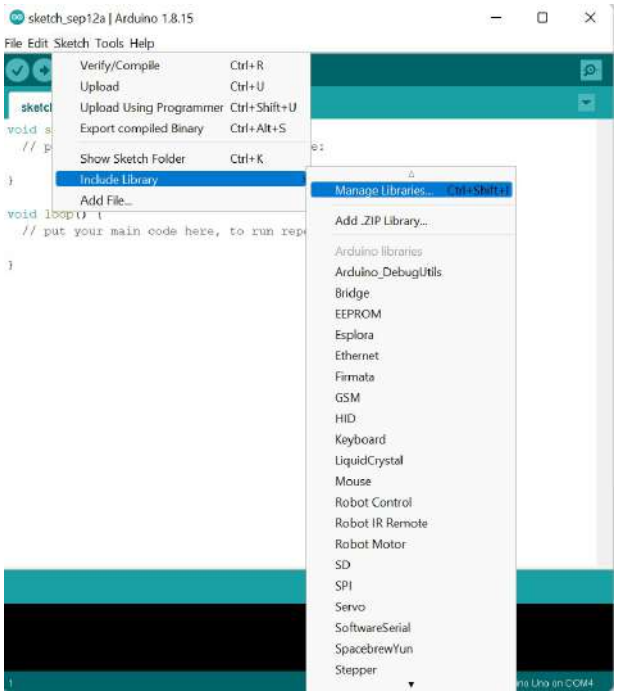


Figura 3.2. Secuencia para instalación de librerías en IDE Arduino.

Se debe esperar que se actualicen las librerías. Luego, escribir en el buscador la librería en particular que se desea instalar. En el caso de los sensores DHT11 o DHT22, se busca Adafruit Unified Sensor y luego instalar, como se muestra en la figura 3.3.

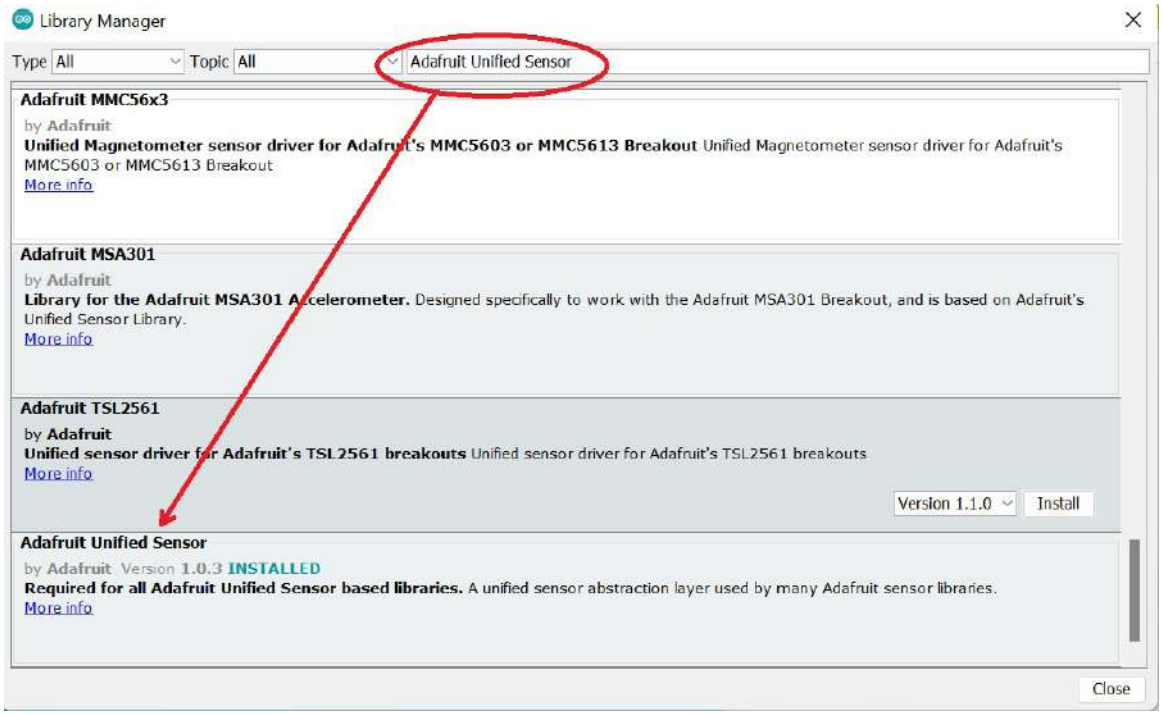


Figura 3.3. Instalando librería Adafruit Unified Sensor para sensor DHT11 o DHT22.

Posteriormente se debe instalar la librería ***DHT sensor library***, la cual de igual manera sirve para controlar el sensor de humedad relativa DHT11 como DHT22 (ver figura 3.4).

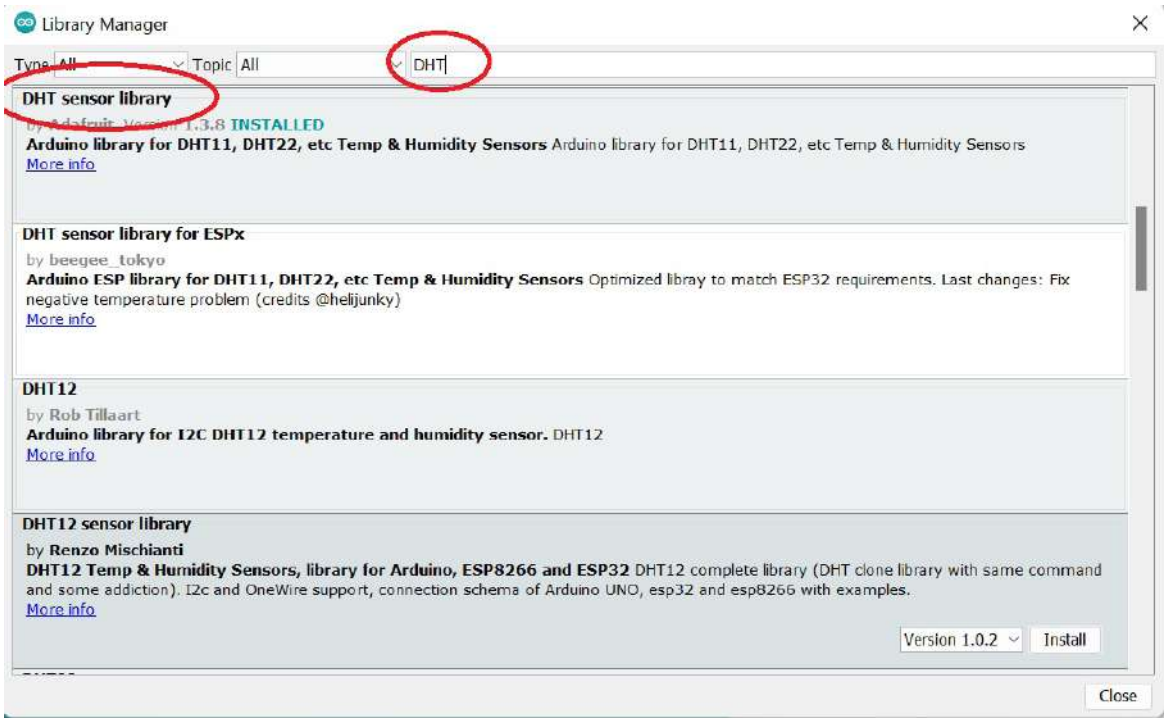


Figura 3.4. Instalando librería DHT sensor library para sensor DHT11 o DHT22.

Luego de instaladas, ambas librerías aparecerán en su IDE Arduino, las que puede verificar desde PROGRAMA → Incluir Librería. En la lista deberán aparecer las librerías instaladas. Entonces ya es posible comenzar con el programa. Lo primero será llamar las librerías que acaba de instalar con el comando #include.

Código Sensor DHTXX

```
#include <Adafruit_Sensor.h>
#include <DHT.h>

// Se define el pin digital donde se conecta el sensor, en este caso será el pin 2
#define DHTPIN 2
// Dependiendo del tipo de sensor será DHT11 o DHT22, en este caso es DHT11
#define DHTTYPE DHT11
 // Se inicializa el sensor DHT11 ya declarado en DHTTYPE, desde el pin 2 declarado en DHTPIN
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  // Se debe inicializar la comunicación en serie
  Serial.begin(9600);
  // Se inicializa el sensor DHT
  dht.begin();
}

void loop() {
    // Se propone esperar 2 segundos entre medidas
  delay(2000);

  // Se definen las variables que leerán humedad relativa (hum) y temperatura (temp)
  float hum = dht.readHumidity();
  // Se lee la temperatura en grados centígrados (por defecto)
  float temp = dht.readTemperature();

// El siguiente comando es opcional para indicar que sale del sensor DHT11 solamente
  {
    Serial.print("Sensor DHT11");
  }

// Se debe notar que hay diferencia entre escribir Serial.println y Serial.print: en uno salta de
// reglón, mientras que en el otro sigue escribiendo en la misma fila.
  Serial.println();
  Serial.print("Humedad: ");
  Serial.print(hum);
  Serial.print(" % ");
  Serial.print("Temperatura: ");
  Serial.print(temp);
  Serial.print(" *C ");
  Serial.println();
}
```

Al implementar este código en el Arduino, lo que realiza es medir, cada dos segundos, la humedad relativa y temperatura del aire (este sensor no mide bajo el agua). Luego muestra los datos obtenidos en el Monitor Serie del Arduino (ver figura 3.5).

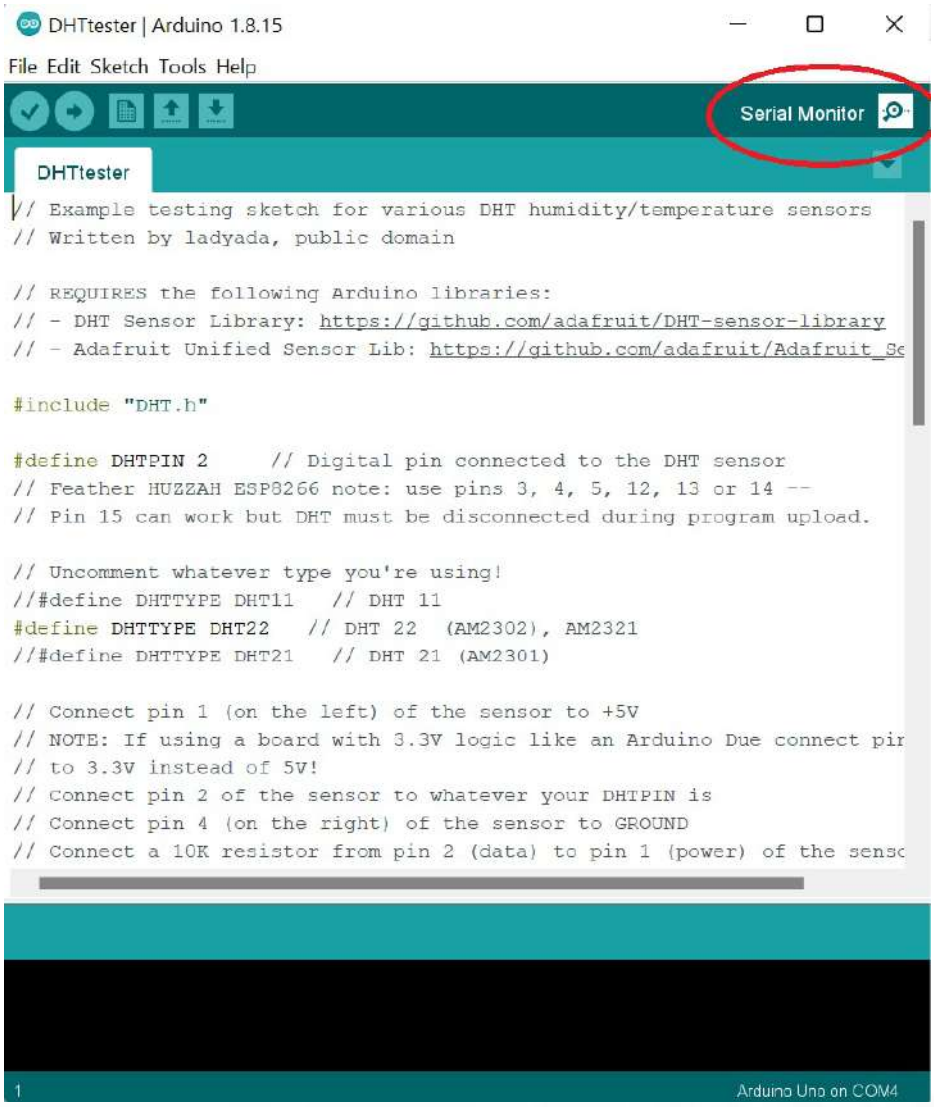


Figura 3.5. Monitor Serie desde donde se observará la medición del sensor DHT11.

TALLER 5

Humedad del SUELO

El control de la temperatura y de la humedad relativa, por ejemplo mediante un equipo de aire acondicionado, permite proporcionar condiciones ambientales óptimas para el ser humano. En el caso de las plantas, también necesitan temperaturas y particularmente cierta humedad del suelo en que se encuentran enterradas.

Componentes

- Protoboard
- Arduino UNO
- Cables
- Sensor YL-69 con módulo LM393

Desarrollo

Para medir la temperatura y humedad relativa del aire hay diversos sensores. Entre los más fáciles de encontrar están los sensores digitales **DHT11** o **DHT22**, vistos en el Taller 3, pero para medir humedad del suelo se necesita un sensor que permita ser introducido en el material para registrar los cambios de resistencia en el suelo. Es decir, cuando se percibe un mayor paso de la intensidad de corriente quiere decir que la humedad aumenta, mientras que cuando ofrece mayor resistencia al paso de la corriente significa que la humedad disminuye.

Uno de los sensores más económicos y sencillos de programar es el sensor **YL-69**. Se trata de una sonda de dos terminales (como un tridente con dos asas) que se introducen en una superficie, y que para llevar los datos al Arduino está compuesto de otro circuito que contiene el **módulo LM393**. Así, el armado base consiste en dos pines del sensor al módulo y luego cuatro pines del módulo al Arduino: alimentación (5V), tierra (GND), D0 (salida digital) y A0 (salida analógica cuyo voltaje es proporcional a la humedad) (ver figura 5.1).

Para hacer la prueba de este sensor y sus medidas, se puede disponer de un vaso con tierra donde tenga ubicado el sensor. De esta manera, puede medir los cambios que experimente el sistema.

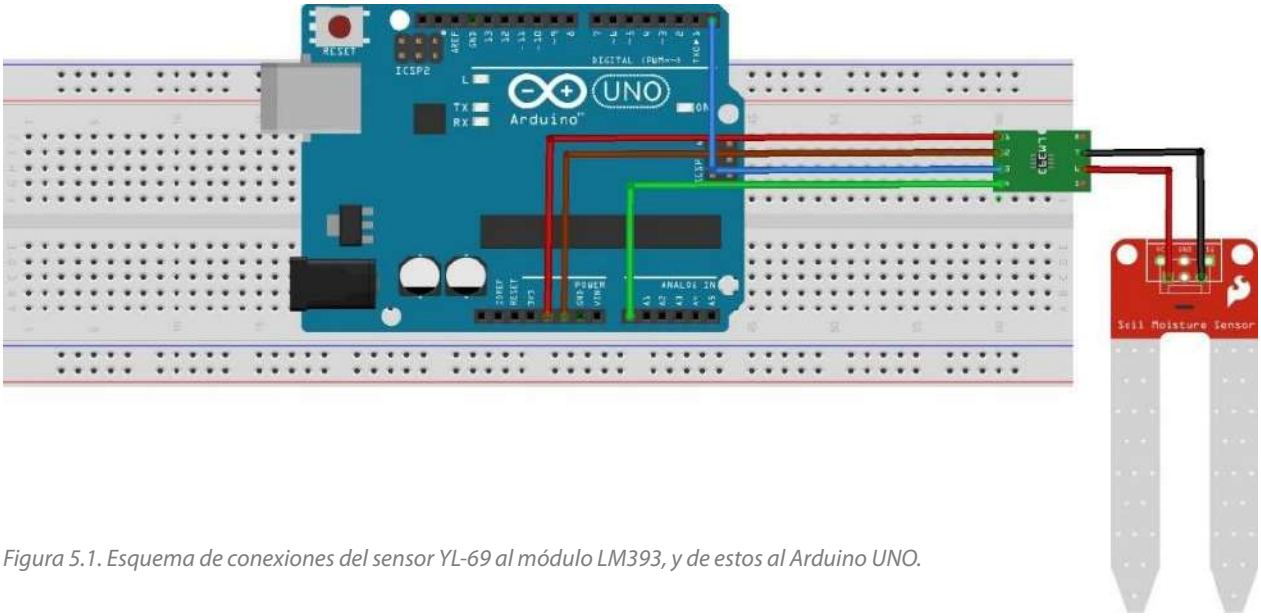


Figura 5.1. Esquema de conexiones del sensor YL-69 al módulo LM393, y de estos al Arduino UNO.

Código sensor YL-69

```
/*
# Este Código de ejemplo conecta el sensor al Arduino en el pin análogo A0
# La interpretación entre la lectura y las condiciones se puede leer
# 0 a 300 suelo seco
# 300 a 700 suelo húmedo
# 700 a 950 suelo mojado
*/

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

void loop() {
Serial.print("Humedad del Suelo:");
Serial.println(analogRead(0));
delay(500);
}
```

Desafío

Calibrar experimentalmente el sensor para que indique medidas porcentuales (cualitativas) de humedad del suelo.

TALLER 6

Radiación **UV**

El wifi, el infrarrojo, el ultravioleta o la luz visible son ejemplos de las múltiples ondas electromagnéticas con las cuales convivimos a diario. Los seres humanos somos capaces de distinguir solo una parte del espectro electromagnético. Esa pequeña parte se conoce como espectro visible, cuyas longitudes de onda están entre los 400 nm y 700 nm, aproximadamente.

Existen rangos con longitudes de onda más largas y otras más cortas, para los cuales necesitamos de instrumentos o sensores que sean capaces de detectarlas. Una parte de esta radiación posee longitudes de ondas más corta y que es más energética, siendo capaz de desplazar electrones de los átomos y modificar su estado. Es decir, puede producir efectos ionizantes.

En uno de los límites del espectro visible, más allá del violeta, se encuentra el espectro ultravioleta (UV) (ver figura 6.1), cuyo rango se encuentra aproximadamente entre los 100 nm hasta los 350 nm. El ultravioleta no es un único valor, ya que posee tres subdivisiones como UVC (100 nm a 280 nm), UVB (280 nm a 315 nm) y UVA (315 nm a 400 nm).

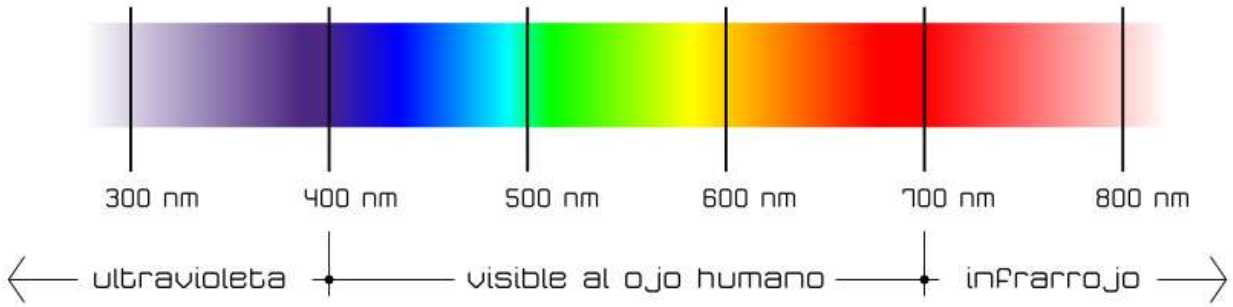


Figura 6.1: Espectro electromagnético visible y extremos.

Índice ultravioleta

Para alertar respecto al impacto que tiene la radiación ultravioleta proveniente desde el sol sobre las personas, existe un índice de radiación UV que sirve como un parámetro de orientación para determinar el riesgo potencial que tiene en la salud. La Organización Mundial de la Salud (OMS) establece un código de colores que asocia los diferentes niveles con un determinador color (ver figura 6.2).

- Verde: nivel bajo, índices 1 y 2
- Amarillo: nivel moderado, índices 3 al 5
- Naranjo: nivel alto, índices 6 y 7
- Rojo: nivel muy alto, índices 8 al 10
- Morado: nivel extremadamente alto, índice 11+

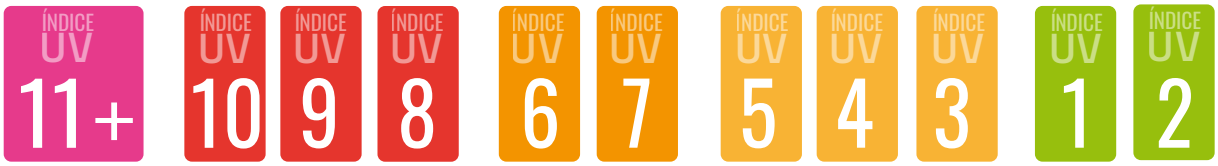


Figura 6.2[8] : Código de colores asociado a nivel de radiación ultravioleta.

8 *Según el instructivo de radiación ultravioleta de origen solar, ACHS.*

Medida de la radiación ultravioleta y cálculo del índice UV

Existen múltiples sensores para medir radiación UV. En este taller se utilizará el sensor UV GUVA S12-SD, sensor de bajo costo capaz de medir un rango de longitudes de onda entre los 240 nm y los 380 nm, siendo adecuado para medir la radiación UVB, que es la más dañina para las personas.

En la hoja de datos del fabricante del sensor (Datasheet) se especifica una tabla con los valores de salida, que puede alcanzar los diferentes niveles del índice UV. De esta manera es posible realizar la calibración del sensor, transformando los valores de diferencia de potencial a índice UV.

Este sensor posee tres pines. Un pin corresponde a la salida de los datos en formato analógico, otro pin es el de alimentación y el tercero debe ir a una de las tierras del Arduino. Puede alimentarse con 3V o 5V, entregando una salida que variará entre 0 y 1200 mV. La lectura en milivolts se debe transformar a los índices de UV (ver figura 6.3).

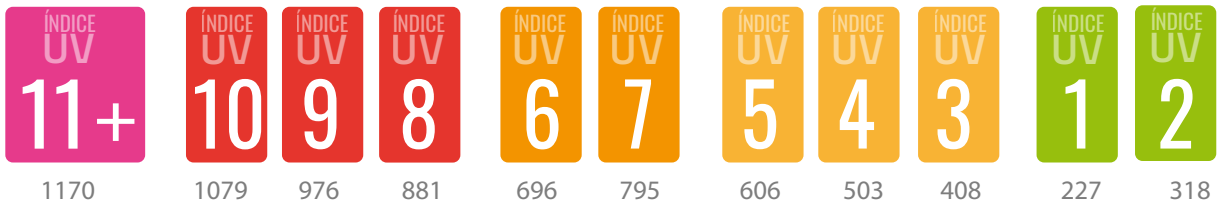


Figura 6.3: Voltaje que representa cada índice UV.

La conexión del sensor GUVA S12 con el Arduino se puede observar en la figura 6.4, conectado el pin análogo al pin A0 del Arduino UNO. La función que permite leer la señal es analogRead() desde el pin A0. Esa lectura es la que se utilizará en la función como sensorValue para finalmente definir el índice UV, con el valor obtenido en sensorVoltage. La ecuación 1 a continuación, esquematiza la relación entre las variables o modelo matemático a trabajar.

$$sensorVoltage = \left(\frac{sensorValue}{1024}\right) * 5 \qquad (1)$$

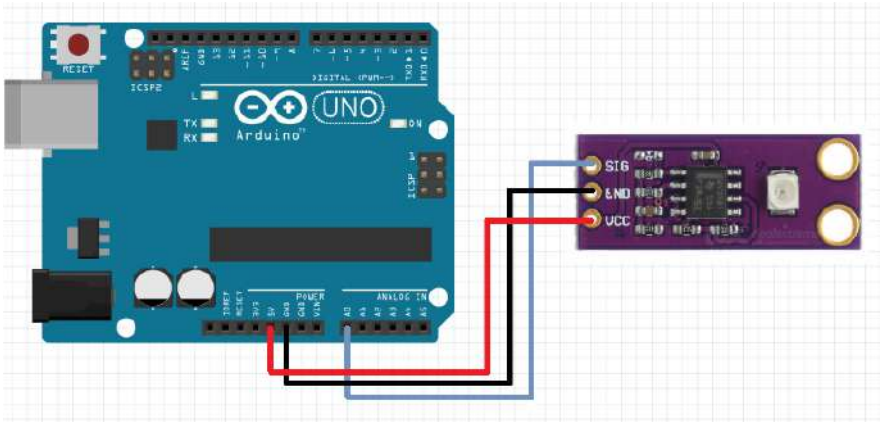


Figura 6.4: Conexión sensor UV GUVA-S12SD con placa Arduino UNO.

Código Sensor UV GUVA-S12SD

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

void loop() {
  float sensorVoltage;
  float sensorValue;
  sensorValue=analogRead(A0);
  sensorVoltage=(sensorValue/1024)*5;
  if(sensorVoltage < 50){
    Serial.println("UV: 0");
    Serial.println("Índice bajo");
  } else if(sensorVoltage >= 50 & sensorVoltage <= 227){
    Serial.println("UV: 1");
    Serial.println("Índice bajo");
  } else if(sensorVoltage >= 228 & sensorVoltage <= 318){
    Serial.println("UV: 2");
    Serial.println("Índice bajo");
  } else if(sensorVoltage >= 319 & sensorVoltage <= 408){
    Serial.println("UV: 3");
    Serial.println("Índice medio");
  } else if(sensorVoltage >= 409 & sensorVoltage <= 503){
    Serial.println("UV: 4");
    Serial.println("Índice medio");
  } else if(sensorVoltage >= 504 & sensorVoltage <= 606){
    Serial.println("UV: 5");
    Serial.println("Índice medio");
  } else if(sensorVoltage >= 607 & sensorVoltage <= 696){
    Serial.println("UV: 6");
    Serial.println("Índice medio");
  } else if(sensorVoltage >= 697 & sensorVoltage <= 795){
    Serial.println("UV: 7");
    Serial.println("Índice medio");
  } else if(sensorVoltage >= 796 & sensorVoltage <= 881){
    Serial.println("UV: 8");
    Serial.println("Índice alto");
  } else if(sensorVoltage >= 882 & sensorVoltage <= 976){
    Serial.println("UV: 9");
    Serial.println("Índice alto");
  } else if(sensorVoltage >= 977 & sensorVoltage <= 1079){
    Serial.println("UV: 10");
    Serial.println("Índice alto");
  } else{
    Serial.println("UV: 11+");
    Serial.println("Índice extremo");
  }
    delay(5000); // Espera 5 segundos en cada lectura
}
```

TALLER 7

{ Medición del nivel de **RUIDO** AMBIENTAL

En esta actividad se pondrá en funcionamiento un sensor de sonido ky-037 (ver figura 7.1) que permite monitorear el nivel de ruido ambiental a través del monitor serial de Arduino.

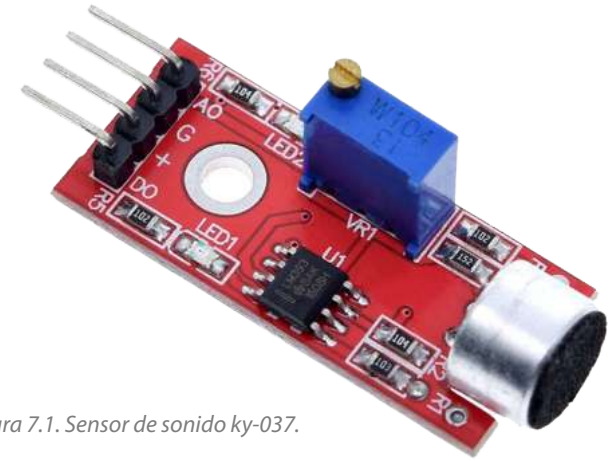


Figura 7.1. Sensor de sonido ky-037.

Conocer el nivel de ruido del ambiente permite tomar decisiones que mejoren nuestra calidad de vida. Según la Organización Mundial de la Salud (OMS), el nivel de ruido recomendado para que no afecte a la salud de las personas es de 65 dB. Si la exposición es superior a 85 dB, ya hay riesgo de pérdida auditiva crónica; si la exposición es frecuente y por encima de 100 dB, hay riesgo de pérdida inmediata de la audición.

La pérdida de audición o hipoacusia inducida por el ruido es un problema de salud que se incrementa con el avance de la civilización. En general, el ruido excesivo se considera como contaminación acústica[9], entre cuyas principales fuentes se encuentran:

- Fuentes fijas: como industrias, discotecas o actividades de construcción, entre otros.
- Fuentes móviles: automóviles, buses, motos
- Ruido de vecinos y actividades en la vía pública
- Actividades como carreteras y aeropuertos, que cuentan con Resoluciones de Calificación Ambiental (Sistema de Evaluación de Impacto Ambiental)

Para la realización de este taller, es necesario disponer de algunos componentes electrónicos y seguir el proceso de construcción que se indica a continuación:

Componentes

- 1 protoboard
- 1 Arduino UNO
- 1 sensor de sonido ky-037
- Cables para conexiones

Desarrollo

En este proyecto se pondrá en funcionamiento el sensor ky-037 con la finalidad de estimar niveles de ruido ambiental. Para lograrlo, se debe armar el circuito que se muestra en la figura 7.2 a continuación.

9 *https://ruido.mma.gob.cl/*

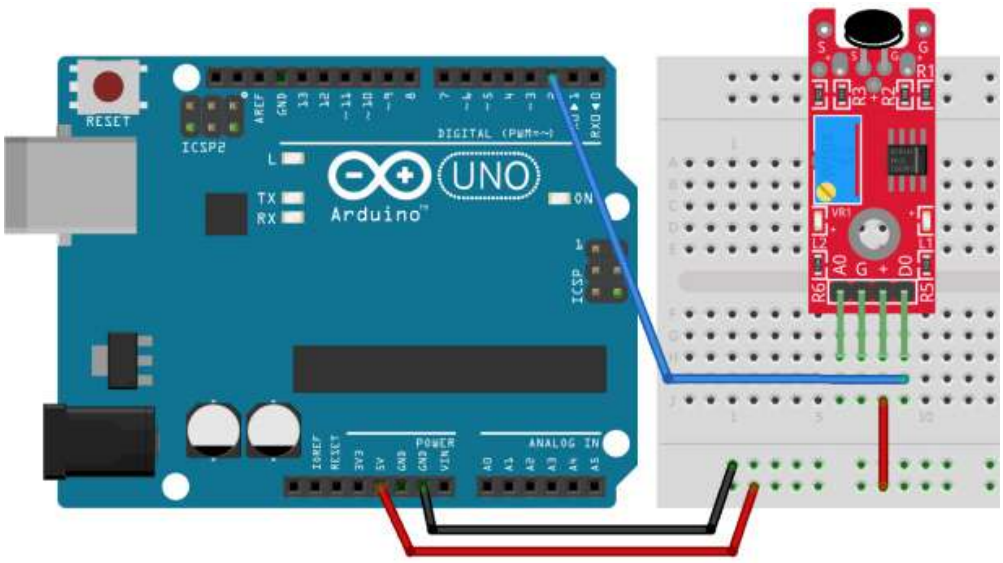


Figura 7.2. Circuito sensor ky-037 usando solo salida digital.

El módulo ky-037 consta de 4 pines (ver figura 7.3): Salida analógica (AO), Tierra (G), Alimentación 5V (+) y Salida digital (DO). Entonces, conecte la salida Digital DO al pin digital 2 de la placa Arduino. A continuación, conecte la línea de alimentación a 5V de Arduino con VCC (+) del sensor. Finalmente, conecte GND de la placa con GND del sensor.

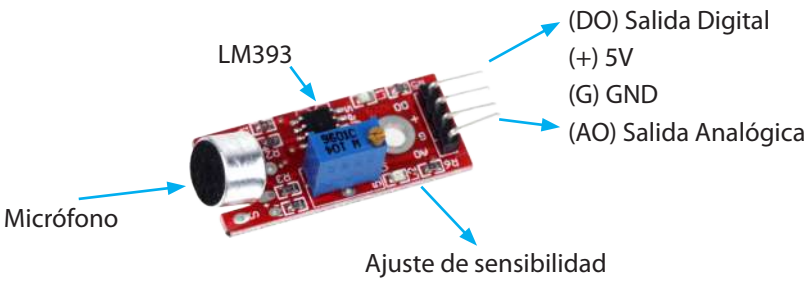


Figura 7.3. Componentes principales del módulo sensor ky-037.

El módulo sensor de sonido ky-037 se basa en el amplificador operacional LM393 y consta de tres componentes relevantes (ver figura 7.3). El primero de estos componentes es el micrófono, que detecta el sonido y lo trasmite como señal eléctrica al amplificador LM393. Este último, como dice su nombre, amplifica la señal que luego se transmite a un circuito comparador, el que apaga la salida y el led conectado a ella si la señal cae por debajo de cierto valor umbral. El esquema del circuito electrónico del módulo ky-037 se muestra en la figura 7.4. El módulo dispone de un potenciómetro que, al girarlo, permite ajustar la sensibilidad del sensor.

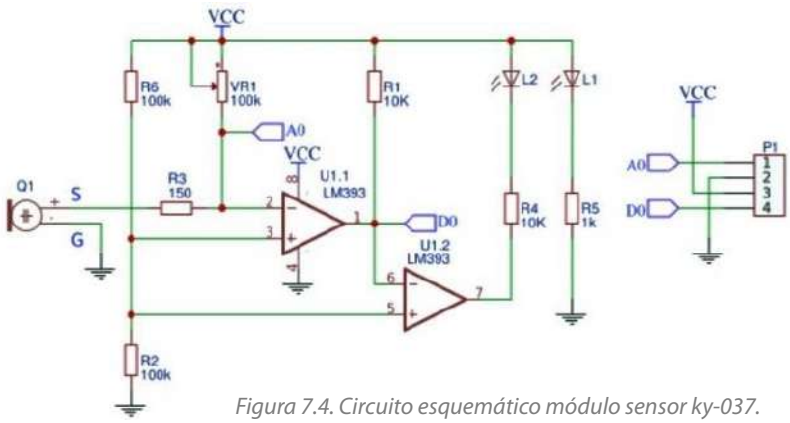


Figura 7.4. Circuito esquemático módulo sensor ky-037.

Código sensor ky-037

```
/* Sonómetro sensor ky-037
 *  utilizando solo salida digital
 *  by labtec
*/

// Definiciones
#define PIN_DIGITAL      2      //DIGITAL

// Variables globales
long veces_leído = 0;
long digital = 0;

// Inicialización
void setup() {
  pinMode(PIN_DIGITAL, INPUT);
  Serial.begin(9600);
}

// Rutina principal
void loop() {

  // Primero se lee el valor del PIN DIGITAL del Sensor de Sonido
  // 50.000 veces y se contabilizan las que superan el umbral
  for (veces_leído = 0, digital = 0; veces_leído <= 50000; veces_leído++) {
    if (!digitalRead(PIN_DIGITAL)) digital++; //lectura digital del sensor de sonido
  }

  Serial.print("DIGITAL: ");
  Serial.println(digital);
 }
```

Desafío

Calibre el sensor para que indique niveles de ruido saludables. Utilice para ello un sonómetro que puede descargar como una aplicación para dispositivos móviles. Compare las lecturas del sensor ky-037 con las mediciones en decibeles de la aplicación, y establezca cuáles lecturas se encuentran dentro de niveles saludables.

TALLER 8

Medición de distancia por **ULTRASONIDO**

En esta actividad se trabajará con un sensor de distancia que trabaja mediante pulsos de ultrasonido. Existen dos tipos de estos sensores en el mercado: un modelo genérico HC-SR04, que consta de 4 pines para conexiones eléctricas, y otro del fabricante PARALLAX. Dada su simplicidad, en este taller se utilizará el sensor genérico HC-SR04.

Cuando hablamos de ultrasonido, nos referimos a una serie de ondas longitudinales y mecánicas que están por sobre el nivel de audición humano. Están presentes, pero no las podemos oír. En distintos tipos de medición no invasivos, estas ondas son útiles principalmente para detectar objetos o medir distancias.

En este taller revisaremos dos tipos de sensor de ultrasonido para la medición de distancias: PING))), con buen alcance y precisión (3 cm a 3 m), y otro de menor costo (HC-SR04), pero que también puede ser aplicado a un amplio rango de sistemas que lo requieran.

Para la realización de este taller, es necesario disponer de algunos componentes electrónicos y seguir el proceso de construcción que se indica a continuación:

Componentes

- 1 protoboard
- 1 Arduino UNO
- 1 sensor ultrasónico PING))) o sensor ultrasónico HC-SR04
- 1 led rojo.
- 1 resistencias de 220 Ω
- Cables para conexiones

Opcionalmente, se pueden usar los modelos disponibles en Tinkercad:

- Sensor HC-SR04: https://www.tinkercad.com/things/3Pe19RRaAwL
- Sensor de PARALLAX: https://www.tinkercad.com/things/1K0K63AEIt4

Desarrollo

En este proyecto se construirá un dispositivo para medir distancia mediante ultrasonido usando el sensor HC-SR04 o el sensor PING))). La principal diferencia entre estos sensores es que el primero cuenta con 4 pines y el segundo con solo 3 pines para conexiones eléctricas. Otra gran diferencia es el precio, siendo más económico el sensor HC-SR04.

Los 4 pines para conexiones eléctricas del sensor HC-SR04 son : Alimentación 5V (Vcc), TRIG (disparador del sonido), ECHO (entrada del sonido reflejado), Tierra (GND) (ver figura 1). Para avanzar en este taller, se debe construir el circuito de la figura 8.1.

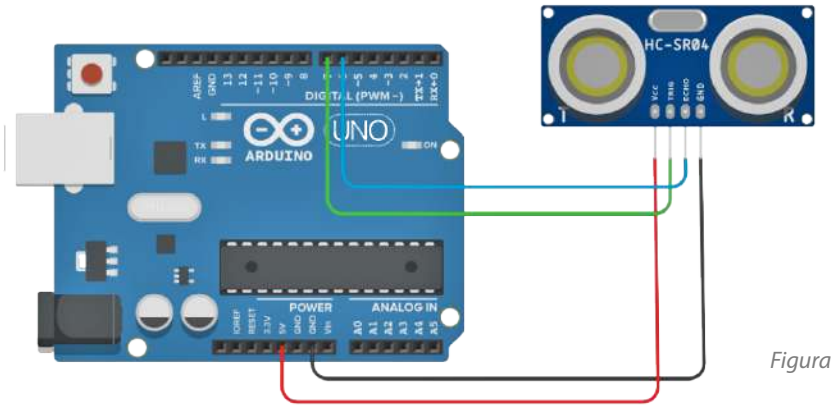


Figura 8.1. Montaje sensor HC-SR04.

Por su parte, el sensor ultrasónico PING))) se caracteriza por disponer de 3 pines para conexiones eléctricas. Dos de estos pines deben conectarse a 5V y GND, mientras que el tercer pin es el único digital (SIG), con el fin de activar una ráfaga ultrasónica y luego "escuchar" el pulso de retorno (ver figura 8.2). Por lo tanto, el sensor PING))) emite y recibe la señal ultrasónica por el mismo pin digital, a diferencia del sensor HC-SR04 que emite por un pin (TRIG) y recibe por otro pin distinto (ECO).

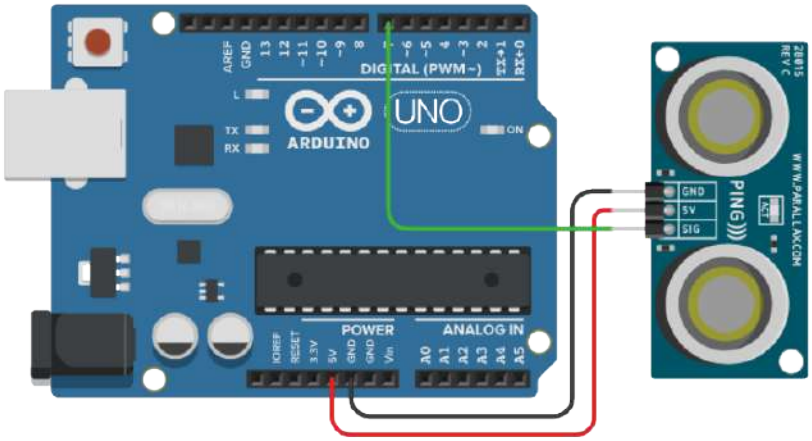


Figura 8.2. Montaje sensor ultrasonido PING))) desde Autodesk® Tinkercad®.

Ambos sensores trabajan mediante el envío de un pulso de sonido ultrasónico que viaja por el medio y se refleja en los objetos que se encuentran en su camino (ver figura 8.3). El sonido, por tanto, viaja desde el sensor y se refleja de regreso, enviando las señales respectivas al microcontrolador.

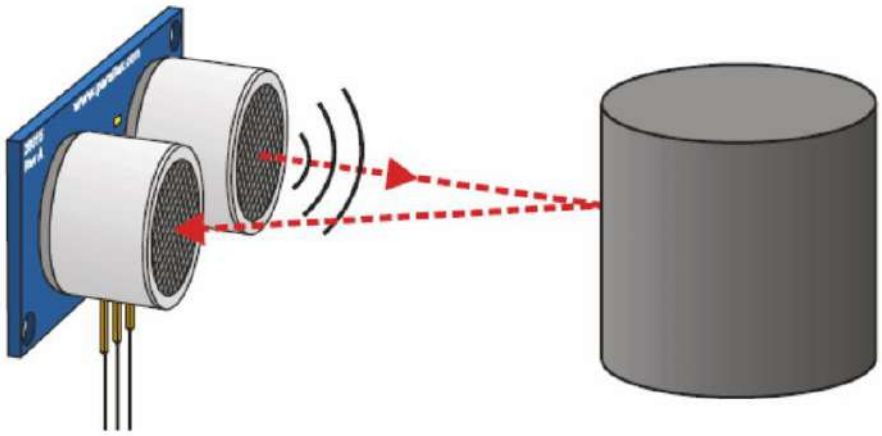


Figura 8.3. Esquema de funcionamiento del sensor ultrasónico.

Puesto que el microcontrolador mide el tiempo de viaje del sonido en el aire, la medicion de la distancia dependerá de la velocidad del sonido en el aire, la cual se se verá afectada por parámetros como la temperatura, entre otros factores del medio.

En general, el microcontrolador Arduino convierte en distancia el tiempo de viaje del sonido en el aire, a partir de la velocidad del sonido en dicho medio. Importante es recordar que la rapidez del sonido en el aire es de 340 m/s a 20°C, o bien, demora 29 microsegundos en recorrer un centímetro, aproximadamente.

El programa a continuación permite medir la distancia y observarla a través del monitor serial. Si se conecta un led al pin 13, el led se encenderá cuando la distancia sea menor de 50 cm. De lo contrario, permanecerá apagado (ver figura 8.4).

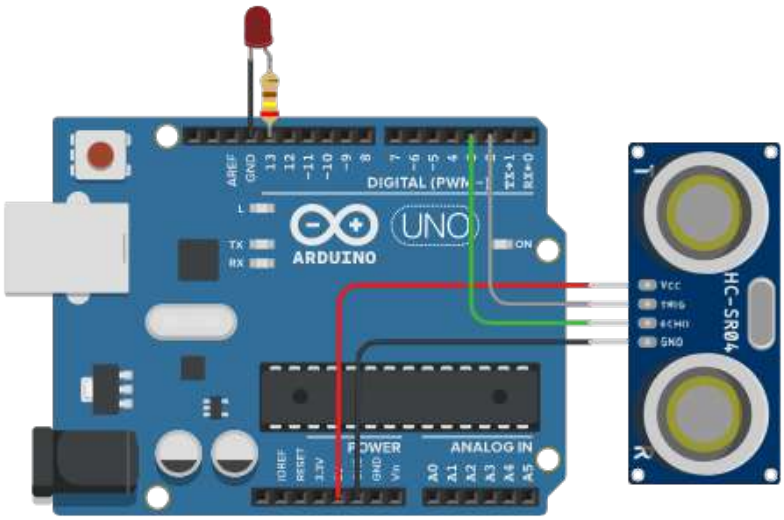


Figura 8.4. Montaje sugerido para trabajar el código HC-SR04 para medir distancia y controlar led.

Código Sensor ultrasónico PING))) (Basado en Tinkercad®)

```
/* Medición de distancia (en cm) mediante ultrasonido
 sensor PING)))
*/

int cm = 0;
long readUltrasonicDistance(int triggerPin, int echoPin) {
pinMode(triggerPin, OUTPUT
digitalWrite(triggerPin, LOW);
delayMicroseconds(2);
digitalWrite(triggerPin, HIGH);
delayMicroseconds(10);
digitalWrite(triggerPin, LOW);
pinMode(echoPin, INPUT);
return pulseIn(echoPin, HIGH);
}
void setup() {
Serial.begin(9600);
}
void loop() {
cm = 0.01723 * readUltrasonicDistance(7, 7);
Serial.print(cm);
Serial.println("cm");
delay(100);
}
```

Código sensor HC-SR04 para medir distancia

```
/* Medición de distancia (en cm) mediante ultrasonido
 sensor HC-SR04
 by LabTec
*/
const int echoPin = 6;
const int trigPin = 7;
const int pausa = 100;
long duración,  cm;

void setup() {
 Serial.begin(9600); //para ver la distancia en monitor serie
 pinMode(13, OUTPUT);
}

void loop() {
  cm = sonar();
    if(cm<50) {
    digitalWrite(13, HIGH);
  }
  else{
     digitalWrite(13,LOW);
  }

  Serial.print(cm); //imprime valores del sensor en monitor serie
  Serial.print("cm");
  Serial.println();
  delay(pausa);
}

//==== Subrutina ====
long sonar()
{
  pinMode(trigPin, OUTPUT);
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(trigPin, LOW);

  pinMode(echoPin, INPUT);
  duración = pulseIn(echoPin, HIGH);
  cm = duración / 29 / 2;  //Aquí se traduce el tiempo de viaje del sonido en distancia.
  return cm;
}
```

Código alternativo Sensor ultrasónico HC-SR04

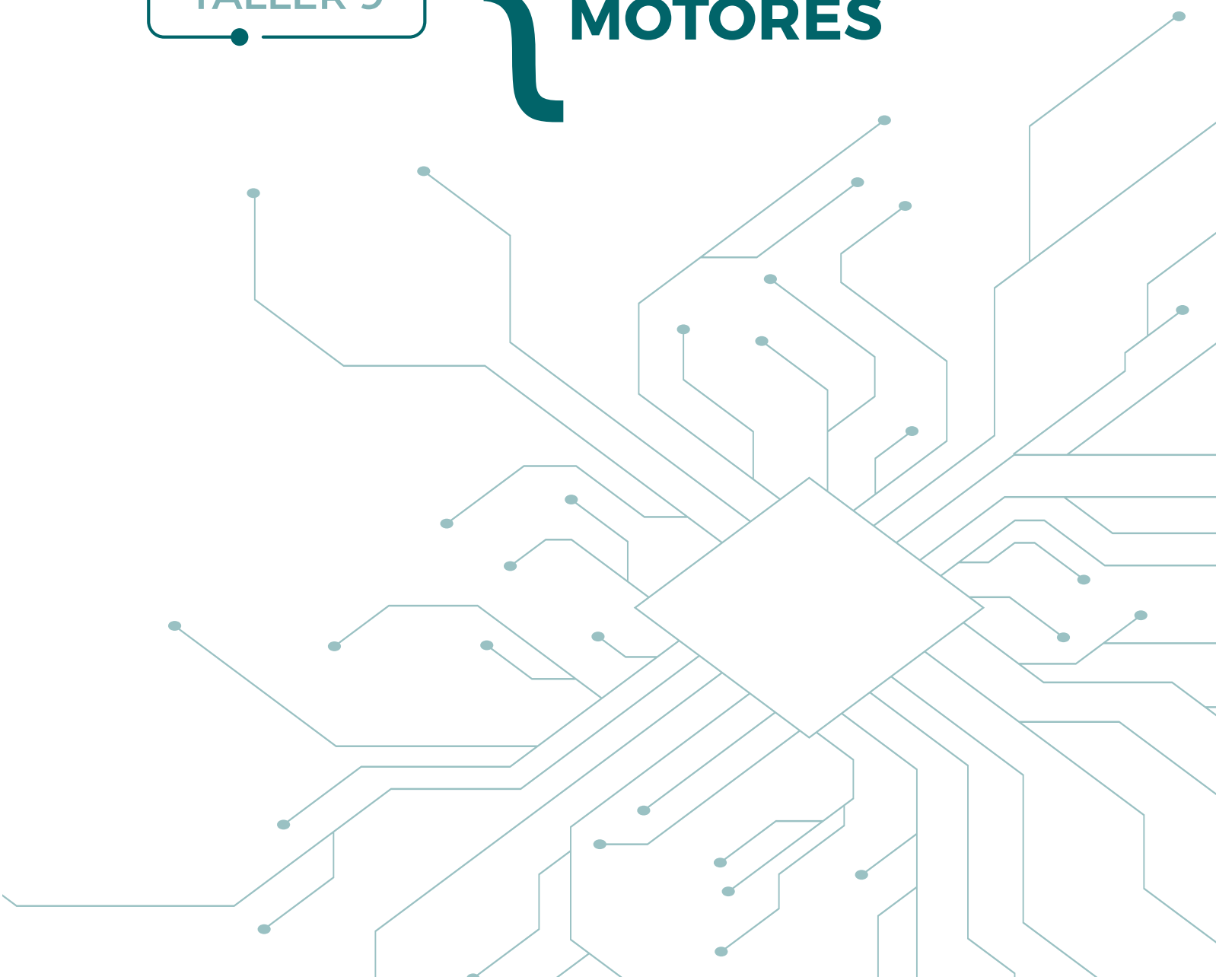
```
/* Medición de distancia (en cm) mediante ultrasonido
   Sensor HC-SR04
   código alternativo
*/

const int Trigger = 7;  // Pin Trigger
const int Echo = 6;     // Pin Echo

void setup() {
Serial.begin(9600); // Se inicializa la comunicación
pinMode(Trigger, OUTPUT); // pin Trigger como salida
pinMode(Echo, INPUT);       // pin Echo como entrada
digitalWrite(Trigger, LOW); // Se inicializa pin digital con 0V
}

void loop() {
long t; // tiempo en ir y llegar el eco
long d; // distancia en centímetros
digitalWrite(Trigger, HIGH);
delayMicroseconds(10);        // Espera de 10 microsegundos
digitalWrite(Trigger, LOW);
t = pulseIn(Echo, HIGH);
d = t/59;           // Se determina la distancia en cm  se divide por 58 o 59??
Serial.print("Distancia:");
Serial.print(d);
Serial.print("cm");
Serial.println();
delay(100);
}
```

Desafío

1. Modifique el programa para que el led parpadee 3 veces si la distancia es menor de 30 cm.
2. Modifique el programa para que el led parpadee 3 veces cuando la distancia sea menor de 50 cm y mayor que 30 cm.

TALLER 9

{ SERVO-MOTORES



En esta actividad se podrán conocer los tipos básicos de servomotores o actuadores, así como el funcionamiento básico de estos dispositivos. Para alcanzar esta meta, es necesario disponer de algunos componentes electrónicos y seguir el proceso de construcción que se indica a continuación:

Componentes

- 1 Arduino UNO
- 1 servomotor de giro continuo
- 1 servomotor de giro posicional o de giro restringido

Opcionalmente, se pueden usar los modelos disponibles en Tinkercad:

- Servo de giro continuo: https://www.tinkercad.com/things/cMSiHyrtKsN
- Servo de giro restringido: https://www.tinkercad.com/things/hLhevC6pIAe

Desarrollo

En este taller se programará un servomotor para que ejecute movimientos básicos. Para lograrlo, se debe armar el circuito que se muestra a continuación (ver figura 9.1). También puedes revisar en Tinkercad los modelos que permiten estudiar el movimiento de los servomotores.

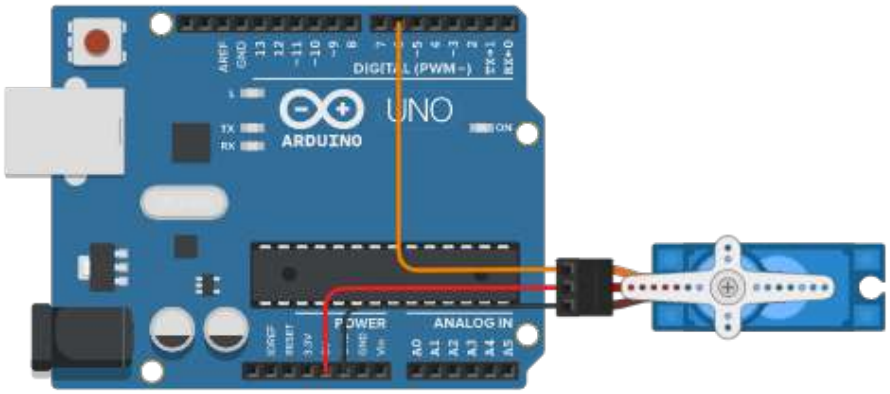


Figura 9.1. Modelo para el funcionamiento básico de un servomotor de giro restringido.

Un servomotor es un dispositivo electromecánico que permite controlar el movimiento en los sistemas robóticos. Está compuesto que tres partes fundamentales (ver figura 9.2):

I) un sistema de control electrónico
II) un sistema mecánico de engranes o de transmisión de movimiento
III) un motor de corriente continua.

El movimiento del servomotor es controlado mediante la duración de los pulsos enviados por el microcontrolador. Este interpreta el sistema electrónico del servomotor, haciendo que motor gire en un sentido u otro, o bien quede en una posición específica.

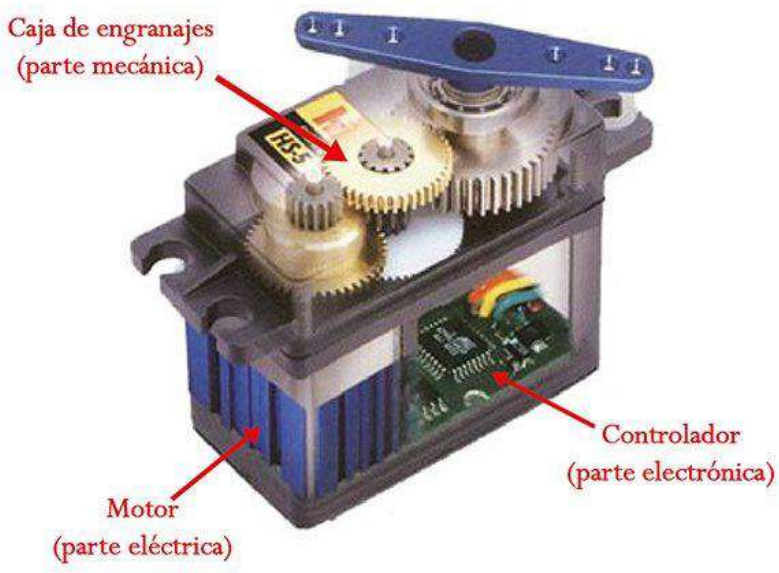


Figura 9.2. Componentes internas de un servomotor.

Existen básicamente dos tipos de servomotores:

I) servo de giro continuo
II) servo posicional o de giro restringido.

Estos servomotores tienen distintas aplicaciones. Por ejemplo, si se requiere construir un robot que se mueva usando ruedas, se deben usar servomotores de giro continuo. En cambio, si se quiere desarrollar una aplicación donde la posición del servomotor debe permanecer en un ángulo preciso, por ejemplo 45°, se recomienda usar un servomotor de giro restringido.

Importante es destacar que los servomotores de giro restringido pueden controlar solo la posición angular del servomotor, entre 0° y 180°. En cambio, los servomotores de giro continuo pueden controlar tanto la rapidez como el sentido de giro, es decir, si el giro es en sentido horario o antihorario, más o menos rápido, e incluso dejarlo detenido.

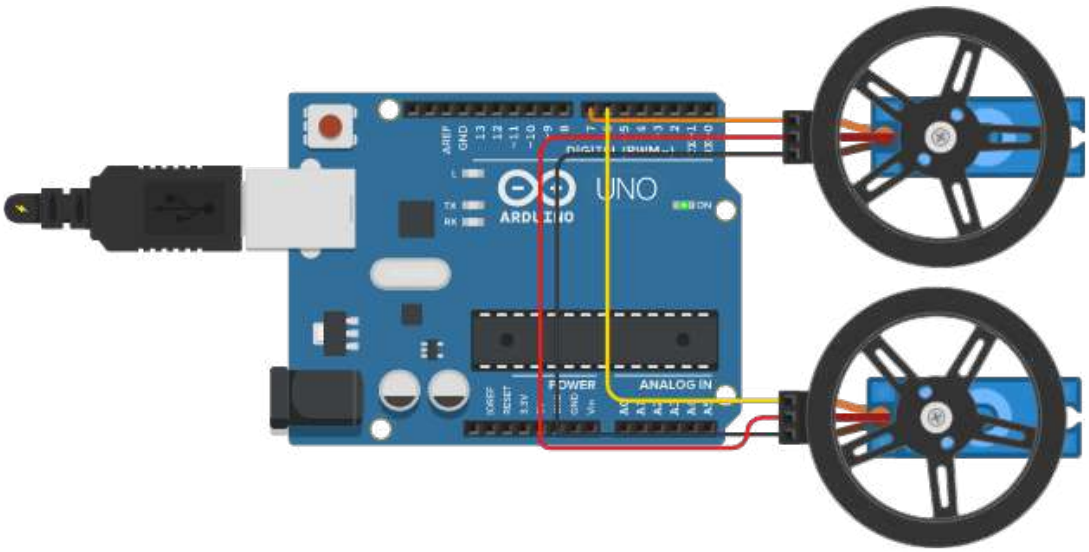
Un servomotor consta de tres cables para conexiones eléctricas, uno para el control del giro y los otros dos para alimentación eléctrica del motor. Tradicionalmente, los colores de los cables son:

I) Negro o café, que se conecta a GND
II) Rojo, que se conecta al positivo de alimentación entre 5 o 6 Volts máximo para un servo estándar
III) Blanco o anaranjado, que se conecta a una salida digital del microcontrolador para el control de movimientos.

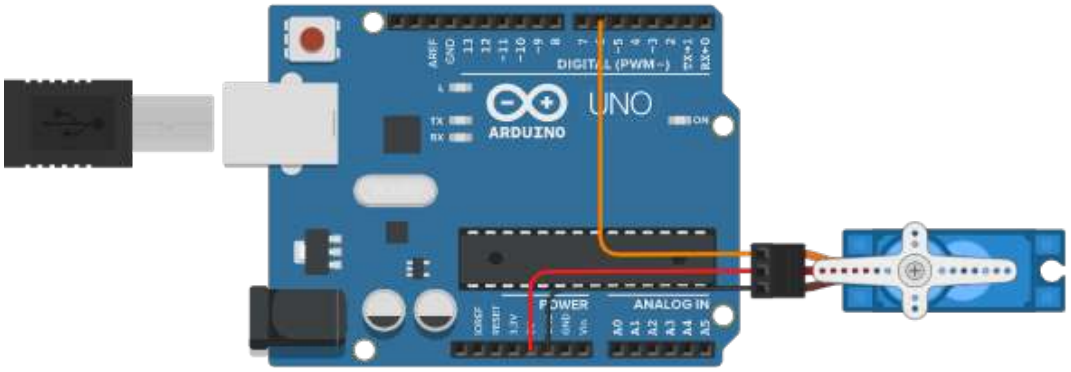
A continuación se muestran dos ejemplos disponibles en Tinkercad. Uno de ellos para el control de un servomotor de giro restringido y el otro para el control de un servomotor de giro continuo:

Ejemplo de códigos para el control de servomotores

- Servo de giro continuo: https://www.tinkercad.com/things/cMSiHyrtKsN
- Servo de giro restringido: https://www.tinkercad.com/things/hLhevC6pIAe




```
/*Control de servomotor giro continuo
   By LabTec
*/

#include <Servo.h>

Servo servo1, servo2;

void setup() {
 servo1.attach(7);
  servo2.attach(6);
 }

void loop() {
  servo1.write(95);//giro antihorario
  servo2.write(85);//giro horario
  delay(100);
 }
```



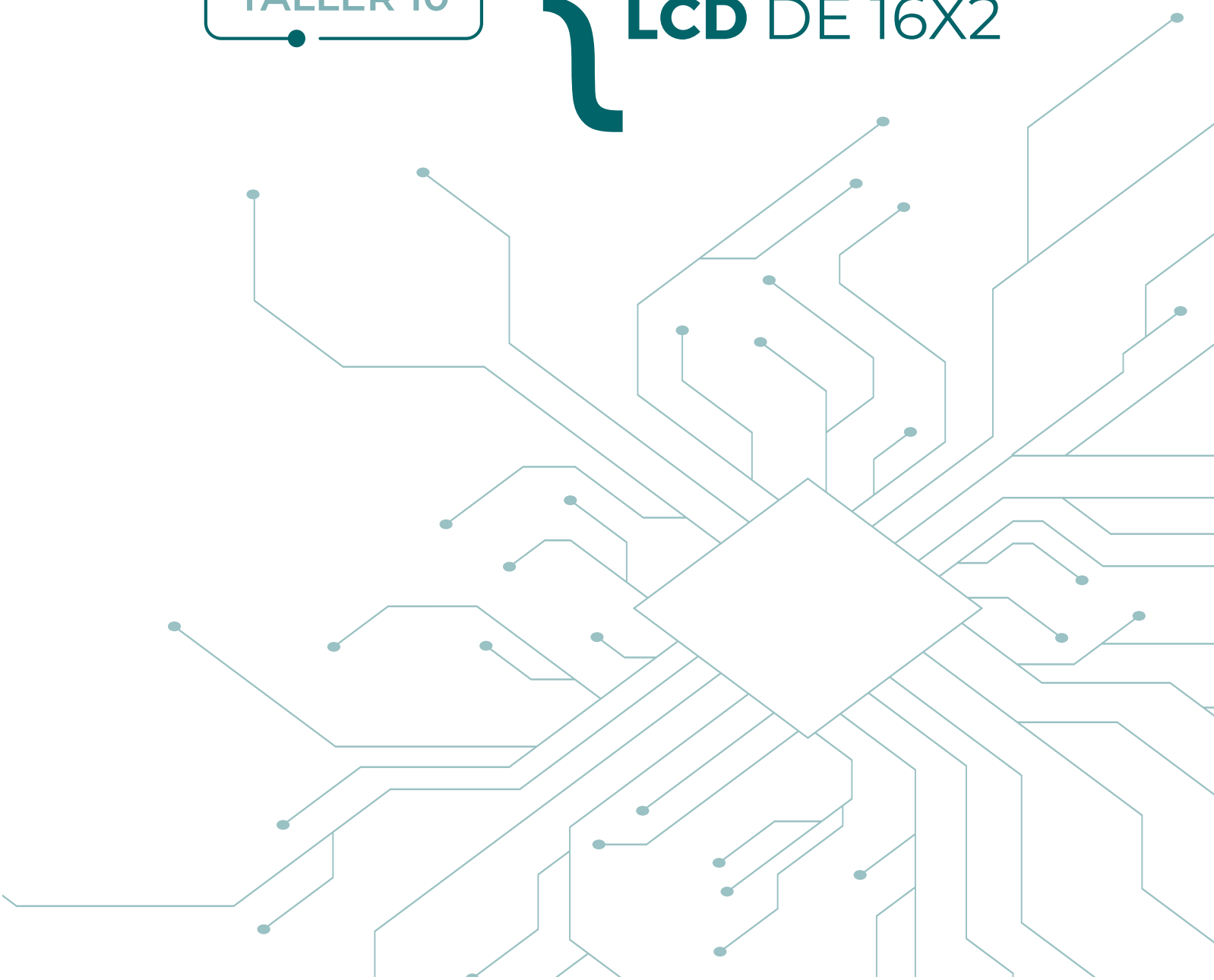

```
/*Control de servomotor giro restringido
   By LabTec
*/

#include <Servo.h>

Servo servo1;
int pos;

void setup() {
 servo1.attach(6);
 }

void loop() {
   for (pos = 0; pos < 180; pos++) {
       servo1.write(pos);
       delay(10);
       }
       for (pos = 180; pos > 0; pos--) {
       servo1.write(pos);
       delay(10);
                  }
}
```

Desafío

Modifique uno de los modelos anteriores agregando un sensor de distancia, por ejemplo, y haga que el motor gire condicionado a las lecturas del sensor. Por ejemplo, que un servo continuo se mueva en un sentido u otro dependiendo de la distancia medida por el sensor.

TALLER 10

Pantalla **LCD** DE 16X2



La programación de un sensor de temperatura, humedad o de otro tipo permite tener un buen conocimiento del funcionamiento de las entradas y salidas (digitales y análogas) del Arduino UNO. Sin embargo, aún se depende del Monitor Serial para visualizar los datos generados por los sensores.

Por esta razón, este taller se centra en tomar control de un display alfanumérico LCD con el microcontrolador Arduino UNO, y comunicar datos a través de esta pequeña pantalla.

Componentes

- 1 display LCD de 16x2 líneas
- 1 Arduino UNO
- 1 potenciómetro de 10kΩ
- Cables para conexiones

Opcionalmente, se pueden usar los modelos disponibles en Thinkercad:

- Display LCD 16x2: https://www.tinkercad.com/things/ir3F79PWZta
- Display LCD 16x2 texto en movimiento: https://www.tinkercad.com/things/iLJ96brDv6v

Desarrollo

Una pantalla LCD (Liquid Crystal Display) es un dispositivo con forma de matriz que se utiliza para mostrar elementos, los que corresponden a algún tipo de salida como datos, mediciones o textos. Las matrices más usuales corresponden a display de 16x1, 16x2, 20x2 o 20x4 líneas. El control de cada uno de los display requiere de al menos 16 cables que controlan la pantalla, los cuales deben ir al Arduino. Es decir, una parte importante de los pines será utilizada para el control del display LCD (ver figura 10.1).

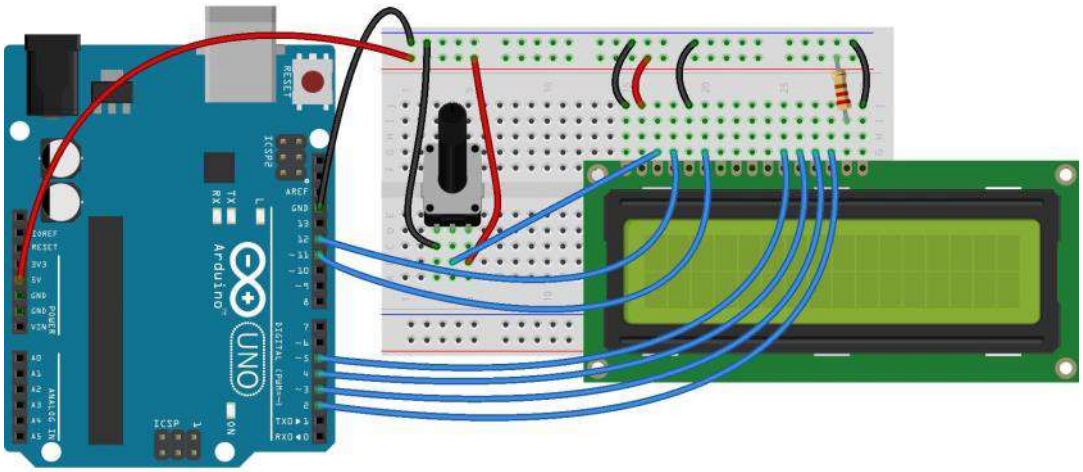


Figura 10.1. Esquema de circuito de display LCD conectado al Arduino UNO.

La pregunta es: ¿si solo en el display se utilizan 16 pines para programar una pantalla que muestre los datos, dónde es posible conectar uno o más sensores?

La respuesta no viene necesariamente de parte de los sensores, independiente si son análogos (valores de 0 a 5V) o digitales (pulsos de unos o ceros). La respuesta viene más bien desde la pantalla del LCD o de un controlador de la pantalla al Arduino. Este controlador está basado en un microcontrolador PCF8574

instalado en una placa PCB llamada bus – I2C. Este dispositivo permite tomar el control absoluto de la pantalla con solo dos pines, más los pines de alimentación (5V) y tierra (GND). Los pines de control corresponden a SDA (datos en la pantalla del display) y SCL (reloj en la pantalla del display), los que se deben conectar dependiendo del tipo de microcontrolador con el que se cuente (ver tabla 10.1).

Tabla 10.1. Conexión pantalla LCD y pines Arduino

ARDUINO	PIN (SDA)	PIN (SCL)
ARDUINO UNO	A4	A5
ARDUINO LEONARDO	2	3
ARDUINO MEGA	20	21

El esquema de conexión del Arduino con la pantalla LCD por medio del bus I2C, se puede observar en la figura 10.2.

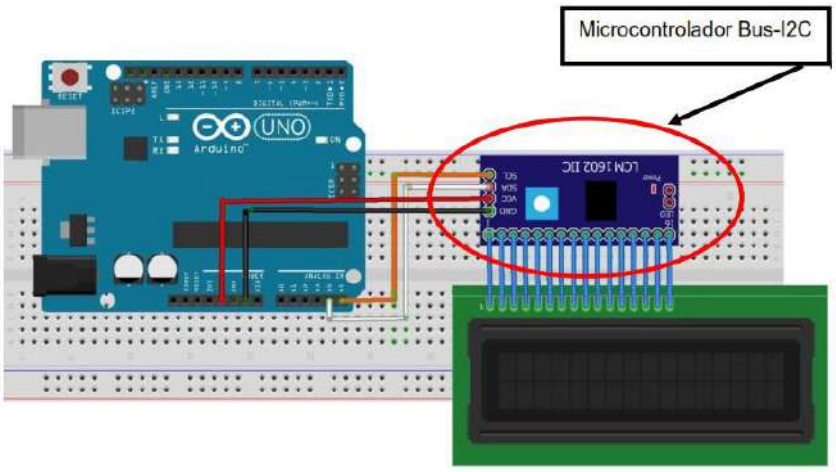


Figura 10.2. Esquema de circuito de display LCD de 16x2 con I2C, conectado al Arduino UNO.

Para el control del display por medio del I2C es necesario instalar la librería LiquidCrystal I2C, para que el Arduino tome el control de la pantalla por medio del bus – I2C. Para ello, desde PROGRAMA espere que se actualicen sus librerías e instale LiquidCrystal I2C como muestra la figura 10.3.

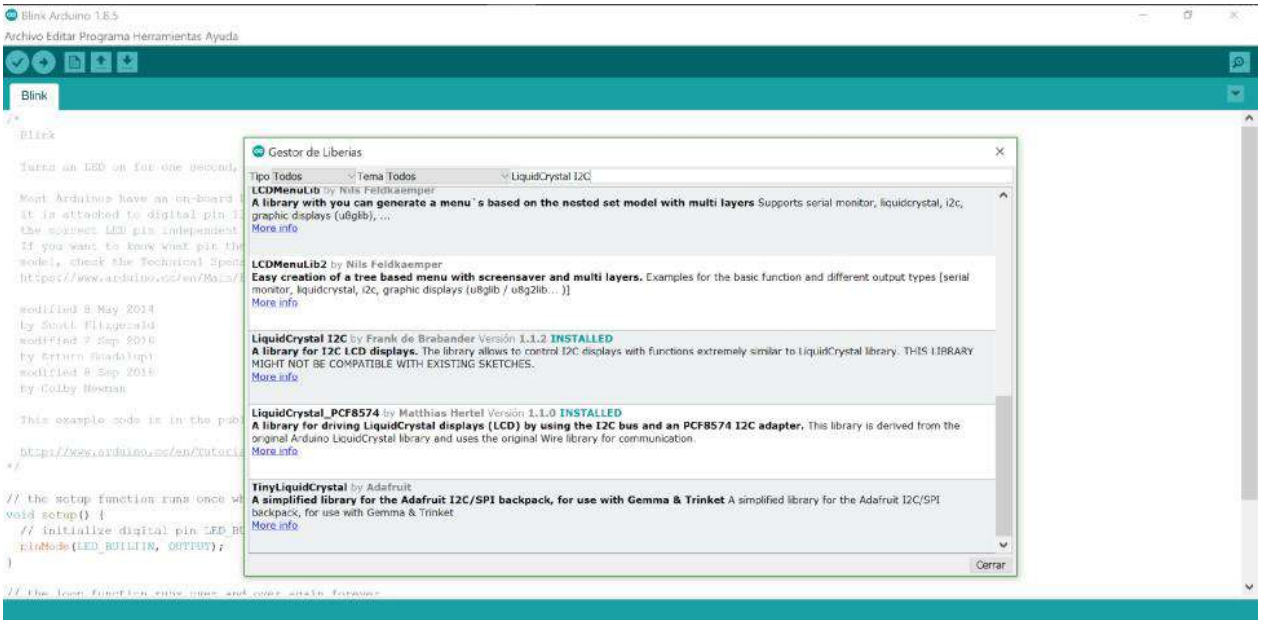


Figura 10.3. Gestor de Librerías de Arduino UNO, búsqueda de LiquidCrystal I2C.

Ya estamos en condiciones de revisar algunos comandos para programar nuestro primer mensaje al mundo desde el LCD.

Funciones básicas para pantalla LCD

lcd.init();
Función de inicio del display LCD.

lcd.backlight();
Enciende la luz de fondo del LCD.

lcd.setCursor(x,y);
Ubica el cursor para escribir en la fila y en la columna x.

lcd.print();
Escribe texto o mensaje en el display LCD, se utiliza de manera similar al comando Serial.print

lcd.clear();
Borra pantalla del display.

scrollDisplayLeft();
Desplaza el contenido de la pantalla un espacio a la izquierda.

scrollDisplayRight();
Desplaza el contenido de la pantalla un espacio a la derecha.

for (int x = 0; x < 10; x++) {
Repite este bucle mientras se cumpla la condición de **x < 10**, incrementando en unidad **x++**
}

Código pantalla LCD 16x2 con I2C

```
//Incluir librerías necesarias para display LCD con bus I2C
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

// La dirección de las librerías puede variar, estas podrían ser 0x27, 0x3F o 0x3A.
// En nuestro caso configura la dirección 0x27 para una pantalla LCD de 16 caracteres y 2 filas.
LiquidCrystal_I2C lcd(0x27,16,2);

// Se inicia comunicación puerto serie
void setup() {
  Serial.begin(9600);
}

void loop() {
lcd.init();                // Inicializa display
lcd.backlight();        // Inicia luz del display, también hay display sin backlight
lcd.setCursor(0,0);   // Sitúa el cursor en la columna 0, en la fila 0.
lcd.print("Hello");   // Todo lo que se escriba entre comillas es textual en el display
lcd.setCursor(0,1);   // Sitúa el cursor en la columna 0, en la fila 1.
lcd.print("World");  // Imprime World en el display
}
```

Anexo

Al escribir el comando para impresión en display, **lcd.print("Hello")**, imprime textualmente: Hello. Si hay una variable obtenida desde algún sensor, como por ejemplo la variable temp, se debe escribir **lcd.print(temp):** en ese caso imprime el valor numérico de la variable.

Suponiendo que en el inicio del programa usted definió una variable **Int a = 2**;

Código en Arduino	Comentario	Salida en display
lcd.print("Hello");	**Imprime el texto entre comillas "Hello"**	**Hello**
lcd.print("a");	**Imprime el texto entre comillas "a"**	**a**
lcd.print(a);	**Imprime el valor de la variable a**	**2**
lcd.print(a); lcd.print("a");	**Imprime el valor de la variable a, seguido imprime el texto entre comillas "a"**	**2a**

TALLER 11

Guardar datos en **TARJETA** SD

Cuando se dispone de un sistema de monitoreo, o sistema de adquisición de parámetros físicos, se vuelve necesario poder guardar los datos para su análisis posterior, particularmente cuando la medición se extiende en el tiempo. En este taller se presenta el mecanismo para guardar datos en un sistema de tarjeta SD, es decir, transformar el Arduino en un Datalogger. Aquí se presentará solo en caso de la tarjeta SD, pero también es posible trabajar con una microSD.

Para la realización de este taller es necesario:

Componentes

- Protoboard
- Arduino UNO
- Cables
- Breakout Board para tarjeta SD
- Tarjeta SD
- Módulo DS1307 o DS3231

Desarrollo

La tarjeta Break Out Board para tarjeta SD posee 10 pines. Al comprarla, poner atención que es necesario comprar los Pin Headers de manera independiente para la conexión al protoboard, pudiendo elegir entre los pin header rectos o en 90° (ver figura 11.1).

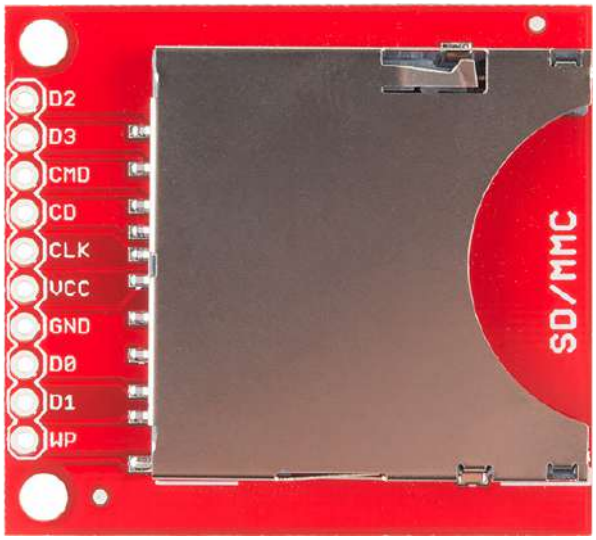


Figura 11.1. Breakout Board para tarjeta SD de Sparkfun.

La librería con la cual trabaja la tarjeta SD o micro SD ya viene con el IDE de Arduino (SD Library), no es necesario instalar la librería como en talleres anteriores. De los 10 pines que posee el Breakout Board para la tarjeta SD, trabajaremos solo con 6 de ellos para la conexión, como muestra la figura 11.2.

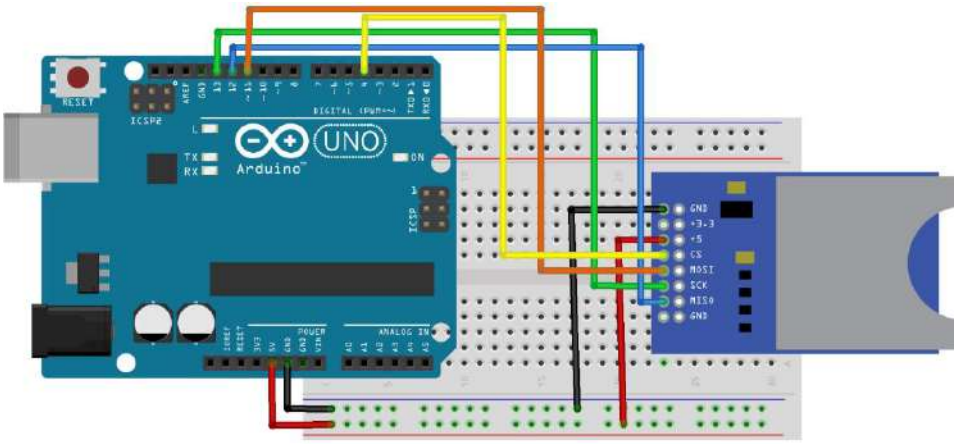


Figura 11.2. Esquema de conexión del Arduino a Breakout Board para tarjeta SD.

En la conexión de la figura 11.2 se ha utilizado un protoboard en el cual las líneas inferiores corresponden a la alimentación de 5V y tierra (GND). Se debe poner atención a que, si bien la placa puede ser alimentada con 3.3V o 5.0V, no se deben conectar ambas a la vez. En la tabla 11.1 se detallan las 6 conexiones propuestas para la Serial Peripherical Interface (SPI) (ver figura 11.2)..

Tabla 11.1. Pines Breakout Board SD y Arduino

Breakout Board SD	Arduino
GND	GND
+5.0V	5V
CS	4
MOSI	11
SCK	13
MISO	12

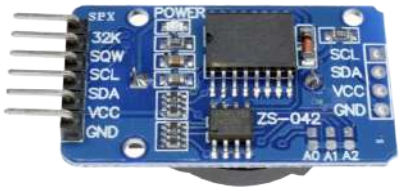
Se utilizan 5V para alimentar el modulo y GND para la tierra. CS se usa para activar la comunicación, MISO corresponde a un pin de transmisión, MOSI se asocia a un pin de recepción y SCK está vinculado al reloj para sincronizar los dispositivos.

Código Datalogger con Tarjeta SD

```
#include <SPI.h>
#include <SD.h>
File myFile;

void setup() {
Serial.begin(9600);
Serial.print("Iniciando Tarjeta SD...");
if (!SD.begin(4)) {
Serial.println("No se puedo inicializar");
return;
}
Serial.println("inicialización correcta");
}

void loop() {
myFile = SD.open("datalog.txt", FILE_WRITE); //se abre el archivo
if (myFile) {
      Serial.print("Escribiendo en Tarjeta SD: ");
      int sensor1 = analogRead(0);
      myFile.print("Tiempo(ms)=");
      myFile.print(millis());
      myFile.print(", sensor1="); // se considera que existe un sensor análogo conectado
      myFile.print(sensor1);
      myFile.close(); // se cierra el archivo

      Serial.print("Tiempo(ms)=");
      Serial.print(millis());
      Serial.print(", sensor1=");
      Serial.print(sensor1);
  }
else {
Serial.println("Error al abrir el archivo");
  }
  delay(500);
}
```

En ocasiones es necesario contar con la fecha, hora, minutos y hasta segundos en que se efectúa la medición. Para ello se propone incorporar un reloj de tiempo real al Arduino (RTC), siendo dos los más comunes y simples de encontrar: DS1307 y DS3231 (ver figura 11.3). En ambos casos, la comunicación es por medio del bus I2C, lo que vuelve bastante sencilla la conexión. Estos módulos necesitan de una batería de litio para ser alimentados: en el caso de DS1307, necesita una pila CR1225, mientras que el DS3231 necesita una pila CR2032.

DS1307



DS3231

Figura 11.3. Relojes de tiempo real (RTC)

Las conexiones al Arduino en ambos casos son similares, por ejemplo:

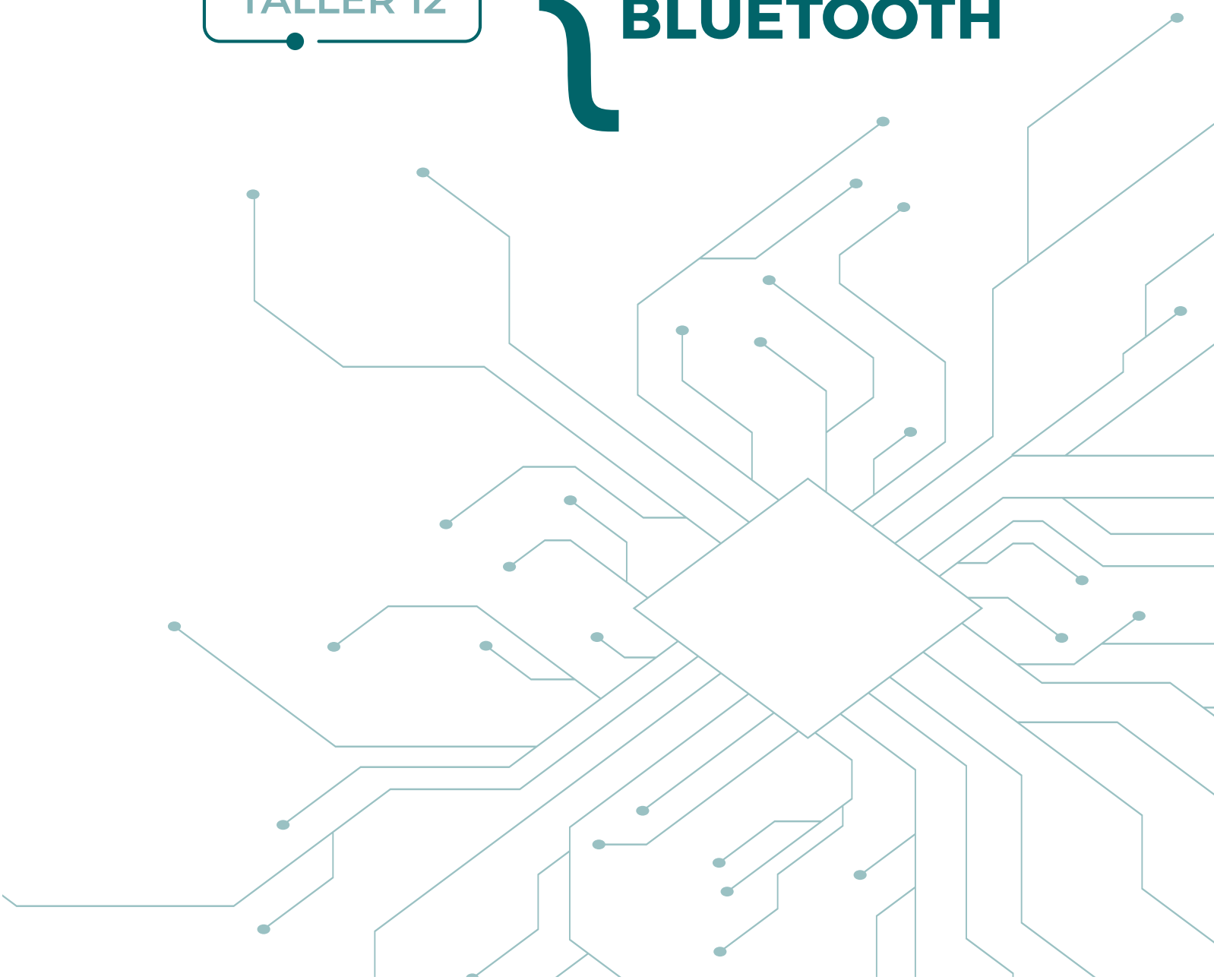
- SCL a A5
- SDA a A4
- Vcc a 5V
- GND a GND

Código Datalogger con RTC

```
#include <SPI.h>
#include <SD.h> // Librerías para la tarjeta SD
#include <Wire.h>
#include "RTClib.h" // Librerías para el RTC
File logFile;
// RTC_DS1307 rtc;  // Depende del RTC a utilizar
RTC_DS3231 rtc;

void setup() {
  Serial.begin(9600);
  Serial.print(F("Iniciando Tarjeta SD..."));
  if (!SD.begin(4))
  {
    Serial.println(F("No se puedo inicializar"));
    return;
  }
  Serial.println(F("inicialización correcta"));
}
// función para simular lectura de sensor
int readSensor() {
  return 0;
}
void logValue(DateTime date, int value) {
  logFile.print(date.year(), DEC);
  logFile.print('/');
  logFile.print(date.month(), DEC);
  logFile.print('/');
  logFile.print(date.day(), DEC);
  logFile.print(" ");
  logFile.print(date.hour(), DEC);
  logFile.print(':');
  logFile.print(date.minute(), DEC);
  logFile.print(':');
  logFile.print(date.second(), DEC);
  logFile.print(" ");
  logFile.println(value);
}
void loop() {
  // Abre el archivo y escribir el valor
  logFile = SD.open("archivo.txt", FILE_WRITE);
  if (logFile) {
    int value = readSensor();
    DateTime now = rtc.now();
    logValue(now, value);
    logFile.close();
  }
  else {
    Serial.println(F("Error al abrir el archivo"));
  }
  delay(1000);  //realiza la muestra cada 1 segundo
}
```

TALLER 12

{ Comunicación por **BLUETOOTH**



Este taller abarca el tema de comunicación desde un punto de vista muy amplio, pues la mayor parte de los dispositivos viene de fábrica con un sistema bluetooth. En el caso de monitoreo con Arduino, permitirá enviar los datos al laptop, a un tablet, al celular, lo cual puede tener ventajas significativas al no tener que conectar una serie de cables extras. En este taller se presentarán los módulos bluetooth para Arduino HC-05 y HC-06.

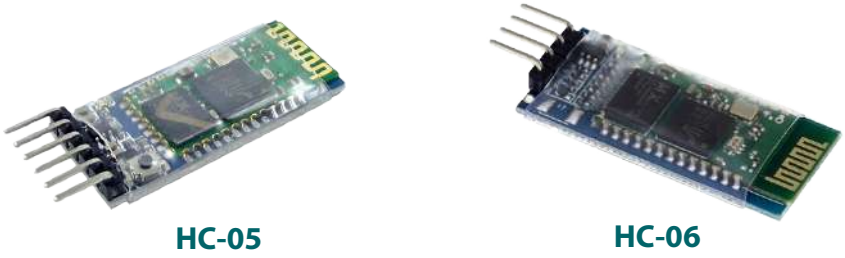
Para la realización de este taller es necesario:

Componentes

- Protoboard
- Arduino UNO
- Cables
- Módulo HC-05 o HC-06

Desarrollo

El módulo HC-05 puede trabajar como maestro y esclavo (en otras palabras, puede recibir e iniciar comunicaciones), a diferencia del módulo HC-06, el cual solo puede trabajar como esclavo, recibiendo comunicación. En aspectos de comunicación, esta diferencia también se traduce en la cantidad de pines que posee cada uno: el HC-05 posee 6 pines, mientras que el HC-06 posee solo cuatro pines. En la figura 12.1 se pueden observar las diferencias físicas entre los módulos HC-05 y HC-06.



HC-05 **HC-06**

Figura 12.1. Módulos bluetooth HC-05 y HC-06.

Los módulos HC-05 y HC-06 requieren el uso de un puerto serie al conectarlo a la placa Arduino. Esto quiere decir que, al cargar un nuevo programa en la placa, se debe desconectar el módulo que se esté utilizando; tras la carga del programa, se puede conectar nuevamente. En el caso del módulo HC-06, el esquema de conexión es como se muestra en la figura 12.2.

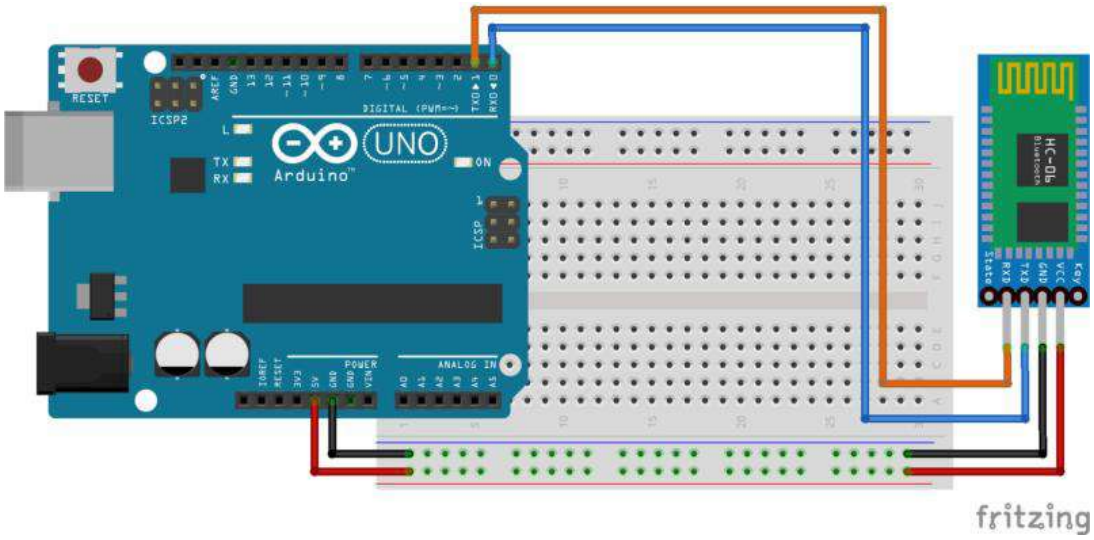


Figura 12.2. Esquema de conexión del Arduino a módulo bluetooth HC-06 de 4 pines.

En el esquema mostrado en la figura anterior solo se han conectado la alimentación 5V, tierra (GND), TX pin de transmisión de datos al pin digital 0 (RX) y RX pin de recepción de datos al pin digital 1 del Arduino (TX).

Código módulo bluetooth HC-06

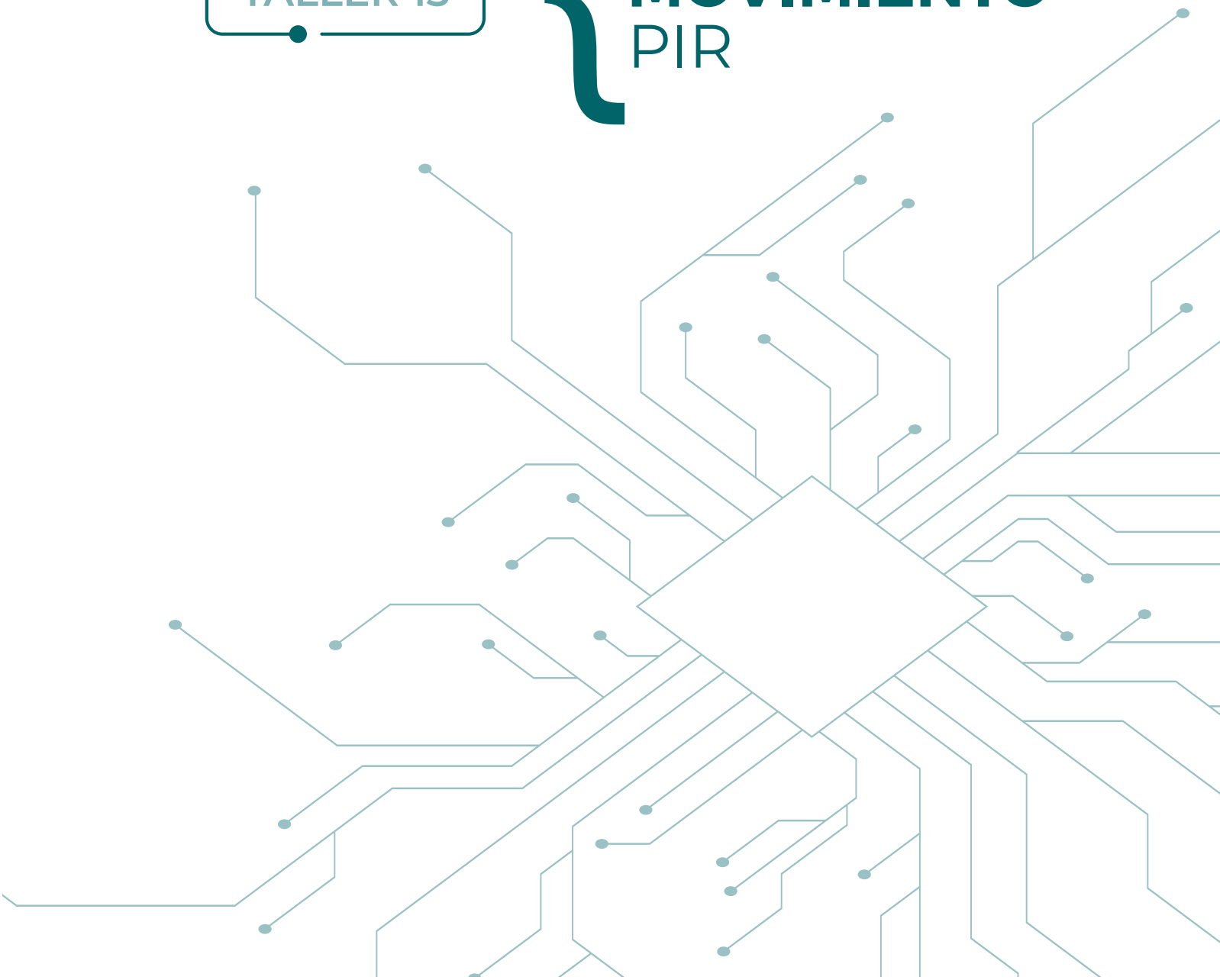
```
/* Código para enviar y recibir datos por bluetooth */

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

void loop() {
  if (Serial.available())
  {
    char caracter=Serial.read();
    Serial.print("Carácter recibido: "); // Acusa recibo de dato
    Serial.println(carácter); // muestra el dato recibido por bluetooth
  }
}
```

Desafío

Pruebe enviar una señal digital compuestas de 1 y 0, con una frecuencia determinada de 1 Hz o 1 señal cada 1 segundo. El desafío es recibir la señal en su celular.

TALLER 13

Sensor de presencia o **MOVIMIENTO** PIR



En esta actividad se pondrá en funcionamiento un típico sensor de presencia o sensor de movimiento PIR (ver figura 13.1).

Los sensores de presencia o movimiento son dispositivos utilizados ampliamente en sistemas de detección de intrusos y sistemas de alarma, ya que se pueden instalar en puertas, ventanas, patios, etc. para avisar de la presencia de intrusos antes de que lleguen al sensor y lo puedan inhabilitar. También se utilizan mucho en el accionamiento de puertas y luces automáticas en zonas comunes de edificios, como entradas o pasillos, y también pueden ser conectados a sistemas de climatización o ventilación.

Para la realización de este taller, es necesario disponer de algunos componentes electrónicos y seguir el proceso de construcción que se indica a continuación.

Figura 13.1: Sensor de presencia o movimiento PIR.

Componentes

- 1 protoboard
- 1 Arduino UNO
- 1 sensor PIR
- 1 resistencias de 220Ω
- 1 led
- Cables para conexiones

Opcionalmente, se puede usar el modelo disponible en Tinkercad:

- Sensor PIR, operación básica: https://www.tinkercad.com/things/6eJ67XopZFJ

Desarrollo

En este proyecto se construirá un sistema de detección de movimiento en base al sensor PIR antes descrito. Este sistema accionará un led al dar cuenta de la detección. El circuito que debe armarse se muestra en la en la figura 13.2. Importante es destacar que, si bien en este caso se activa un led cuando el sensor detecta movimiento, el sistema (hardware y software) puede ser modificado para accionar cualquier otro dispositivo externo, como alarmas sonoras, luces de mayor potencia, etc.

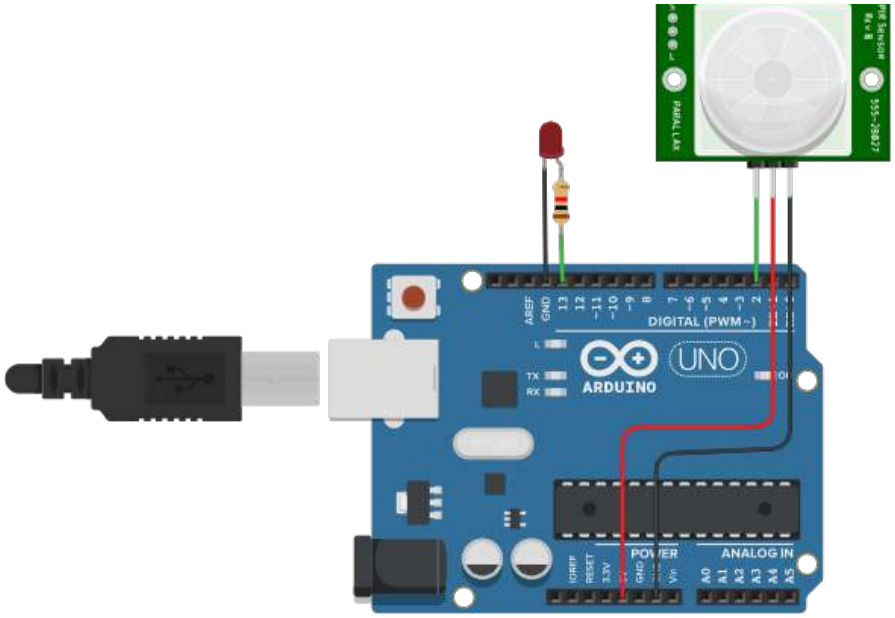


Figura 13.2. Circuito sensor PIR para activar un led

Código que controla el sensor PIR

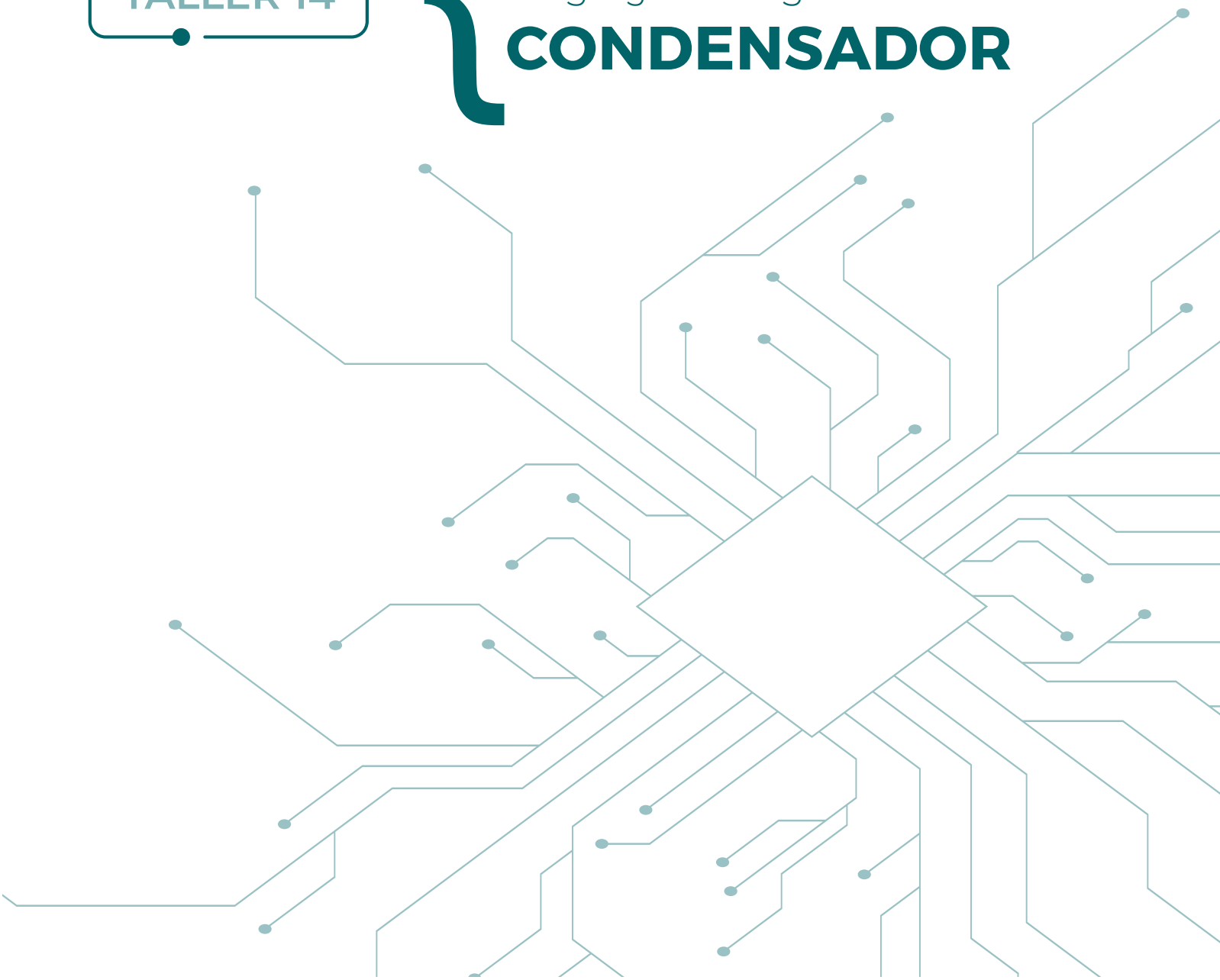
```
/* Sensor PIR: operación básica
   by LabTec
   labtec@umce.cl
*/

const int LEDPin= 13;
const int PIRPin= 2;

void setup() {
 pinMode(LEDPin, OUTPUT);
 pinMode(PIRPin, INPUT);
}

void loop() {
 int value = digitalRead(PIRPin);

 if (value == HIGH)  {
  digitalWrite(LEDPin, HIGH);
  delay(50);
  digitalWrite(LEDPin, LOW);
  delay(50);
 }
 else  {
  digitalWrite(LEDPin, LOW);
 }
}
```

Desafío

Modificar el sistema para que detecte un objeto y active un motor o alarma sonora (por ejemplo: un Buzzer piezoeléctrico).

TALLER 14

Sensor de luz RC-LDR mediante carga y descargar de un CONDENSADOR



En esta actividad se pondrá en funcionamiento un sensor de luz para detectar cambios en la intensidad luminosa ambiental.

Los sensores de luz se pueden utilizar para encender una luz al llegar la noche o detectar líneas para que las siga un robot mientras navega. En particular, en este taller se mostrará cómo detectar cambios en la intensidad luminosa mediante lecturas digitales en un sensor, basado en la carga y descarga de un condensador.

Para la realización de este taller, es necesario disponer de algunos componentes electrónicos y seguir el proceso de construcción que se indica a continuación.

Componentes

- 1 protoboard
- 1 Arduino UNO
- 1 fotorresistencia LDR
- 1 resistencias de 220Ω
- 1 condensador cerámico de 0.1µF
- Cables para conexiones

Opcionalmente, se puede usar el modelo disponible en Tinkercad:

- Sensor de luz RC-LDR: https://www.tinkercad.com/things/g1z2z8B3Mcf
- Sensor de luz RC que activa un led: https://www.tinkercad.com/things/b9T38mYL4pq

Desarrollo

En este proyecto se construirá un sistema de detección de luz que permite conocer cómo varía la intensidad de luz en el ambiente. Para lograrlo, se debe construir el circuito que se muestra en la figura 14.1. Es importante destacar que este sensor se basa en un circuito RC, es decir, el nivel de luz es traducido a una medida proporcional al tiempo de carga o descarga del condensador, la cual se observa a través del monitor serial de Arduino. Por lo tanto, los componentes principales de este sensor son la fotorresistencia LDR, que detecta el nivel de luz, y el condensador cerámico, que se carga y descarga en un proceso que se repite sistemáticamente.

Este hecho implica que la detección del nivel de luz no tiene un comportamiento lineal sino más bien exponencial, de acuerdo al principio de carga y descarga de condensadores. En otras palabras, este taller permite desarrollar nociones referidas a aplicaciones de funciones no lineales en procesos de control de dispositivos. También permite estudiar el proceso mismo de carga y descarga de condensadores.

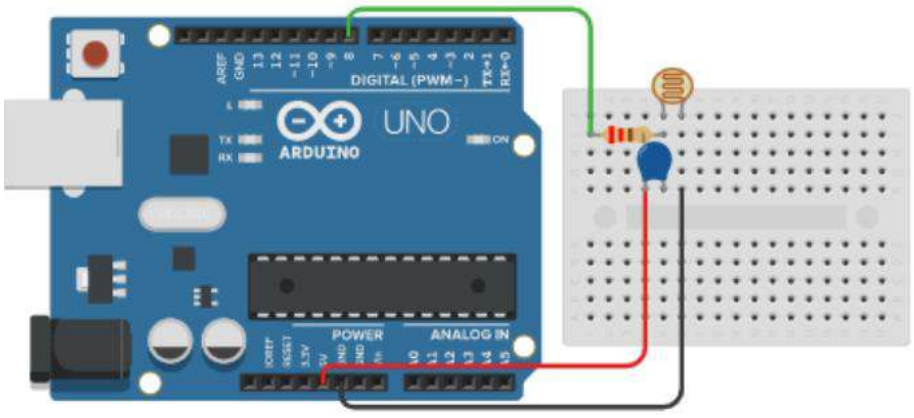


Figura 14.1. Circuito sensor PIR para activar un led.

Código que controla el sensor de luz RC-LDR

```
/* sensor RC-LDR
   by LabTec
   labtec@umce.cl
*/

//=== Estructura principal ===
int sensorPin = 8;
long sensorRC = 0;

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

void loop() {
 sensorRC = RCtime(sensorPin);
 Serial.println(sensorRC);
 }

//=== Subrutinas ===
long RCtime(int sensPin) {
  long result = 0;
  pinMode(sensPin, OUTPUT);
  digitalWrite(sensPin, HIGH);
  delay(1);

  pinMode(sensPin, INPUT);
  digitalWrite(sensPin, LOW);
  while(digitalRead(sensPin)){
   result++;
  }

  return result;
}
```

Desafío

Modificar el sistema para que active un led cuando el nivel de oscuridad supere cierto umbral Por ejemplo, que se active un led cuando se apagan las luces de la habitación donde se ubica el sensor.

TALLER 15

Sensor de luz
ANALÓGICO

Al igual que el taller 14, en esta actividad se pondrá en funcionamiento un sensor de luz para detectar cambios en la intensidad luminosa ambiental, así como para detectar líneas negras que pueda seguir un robot mientras navega por una pista plana.

Los sensores de luz se utilizan en diversas aplicaciones como encender una luz al llegar la noche, detectar líneas para que las siga un robot mientras navega, etc. En particular, en este taller se mostrará cómo detectar cambios en la intensidad luminosa mediante el uso de lecturas analógicas con Arduino.

Para la realización de este taller, es necesario disponer de algunos componentes electrónicos y seguir el proceso de construcción que se indica a continuación.

Componentes

- 1 protoboard
- 1 Arduino UNO
- 1 fotorresistencia LDR
- 1 resistencia de 1kΩ
- Cables para conexiones

Opcionalmente, se puede usar el modelo disponible en Tinkercad:

- Sensor de luz analógico: https://www.tinkercad.com/things/18RWvTHbDiY

Desarrollo

En este proyecto se construirá un sistema de detección de luz que permite conocer cómo varía la intensidad de luz en el ambiente. Para lograrlo, se debe construir el circuito mostrado en la figura 15.1. Es importante destacar que este sensor se basa en un circuito con dos resistencias en serie. La fotorresistencia es el dispositivo que detecta los cambios en la intensidad luminosa, y esos cambios afectan al valor de la resistencia misma. Dichos cambios producen un ajuste lineal en la diferencia de potencial o voltaje entre los extremos de la fotorresistencia.

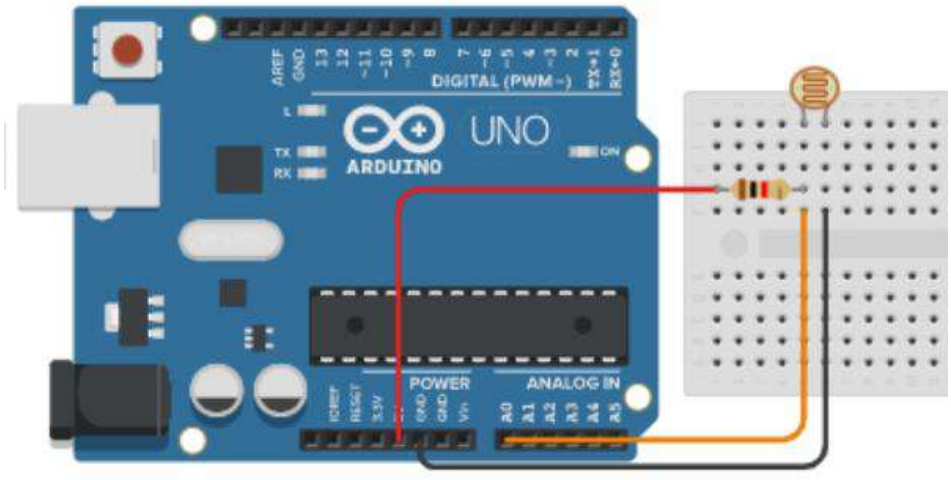


Figura 15.1. Circuito sensor PIR para activar un led.

Código que controla el sensor de luz analógico

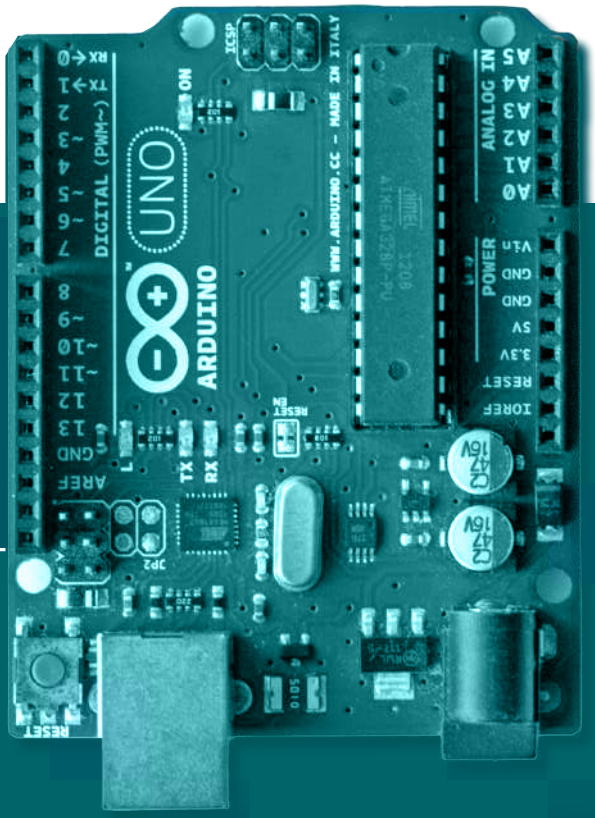
```
/*sensor de luz LDR análogo
  by LabTec
  labtec@umce.cl
*/

//=== Estructura principal ===
int p1Analog = 0;     //analog pin

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

void loop()  {
 long analogLDR = analogRead(p1Analog);

 Serial.print("Análogo: ");
 Serial.println(analogLDR);

 delay(250);

}
```

Desafío

Modifica el sistema para que active un led cuando el nivel de oscuridad supere cierto umbral. Por ejemplo, que se active un led cuando se apagan las luces de la habitación donde se ubica el sensor.

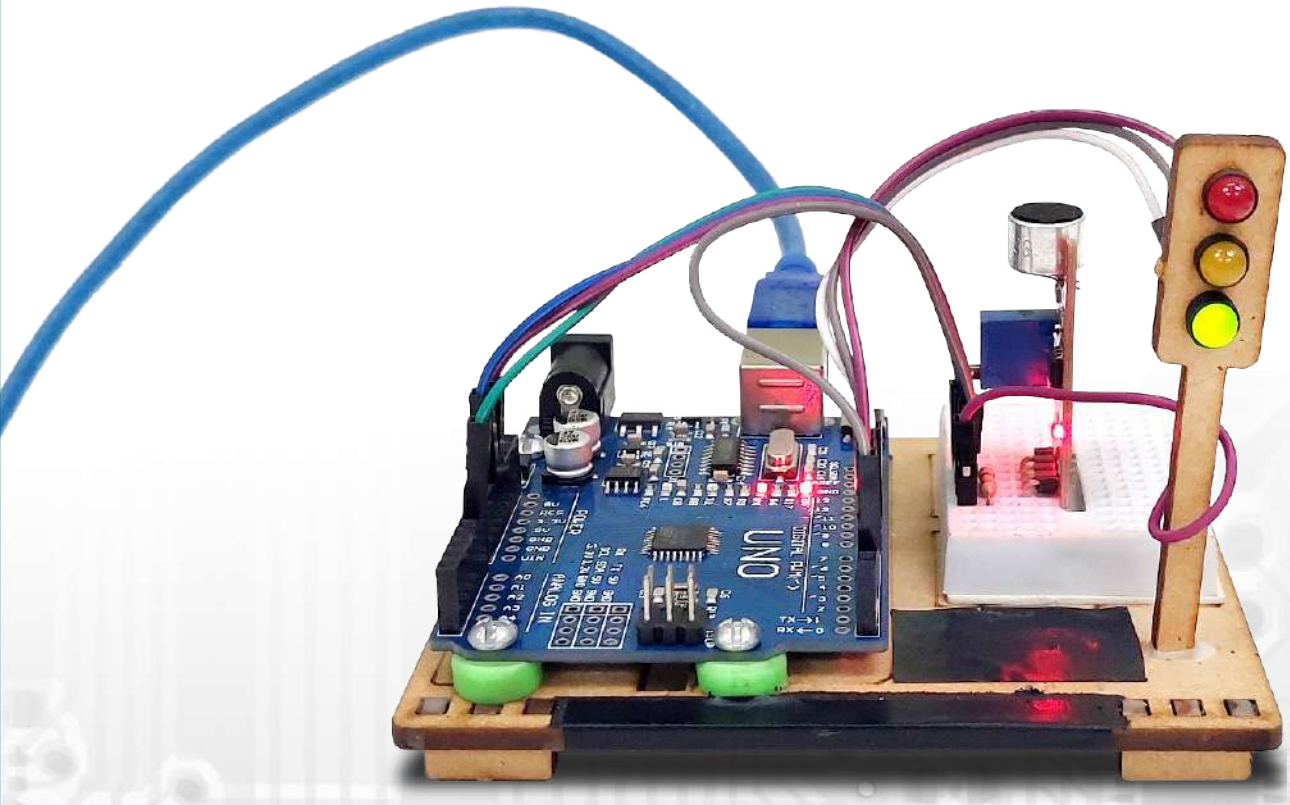
SECCION II

PROYECTOS



PROYECTO 1

Construcción y operación de un **sonómetro semáforo**



En esta actividad se puede construir un semáforo que muestre distintos niveles de intensidad sonora. A partir del ajuste y calibración del sensor de sonido en el ambiente donde se instalará, se pueden establecer los niveles óptimos para que el semáforo muestre:

I) Luz verde: cuando el volumen del sonido sea adecuado
II) Luz amarilla: cuando el volumen del sonido sea ligeramente alto pero tolerable
III) Luz roja: cuando el nivel de intensidad sea alto.

Para la realización de este taller, es necesario disponer de algunos componentes electrónicos y seguir el proceso de construcción que se indica a continuación:

Componentes

- 1 protoboard
- 1 Arduino UNO
- 1 módulo sensor de sonido ky-037
- 3 led: rojo, amarillo y verde.
- 3 resistencias de 220Ω (una para cada led)
- Cables para conexiones

Actividades previas relacionadas

- **Taller 3:** Construcción de un semáforo con leds
- **Taller 7:** Medición del nivel de ruido ambiental

Desarrollo

En este proyecto de sonómetro con semáforo indicador del nivel de sonido, deberás armar el circuito que ser muestra a continuación en la figura 1.

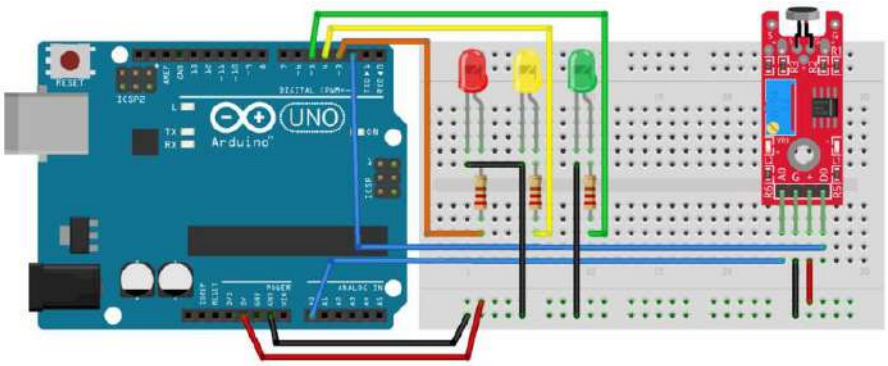


Figura 1. Circuito sonómetro con semáforo.

Para construir el circuito final debes seguir las instrucciones, que dividiremos en dos fases:

Fase 1

Coloque el módulo sensor de sonido en la protoboard y conéctelo a la placa Arduino. Usar como referencia la figura 1. El circuito que corresponde al sensor ky-037 lo puedes revisar en el Taller 7 de este libro. Conéctese la salida analógica AO al pin analógico A0 de la placa Arduino, luego la salida Digital DO al pin digital 2 de la placa. A continuación, conecte la línea de alimentación a 5V de Arduino con VCC (+) del sensor, y finalmente conecte GND de la placa con GND del sensor. En la figura 2 se muestran las partes principales del módulo.

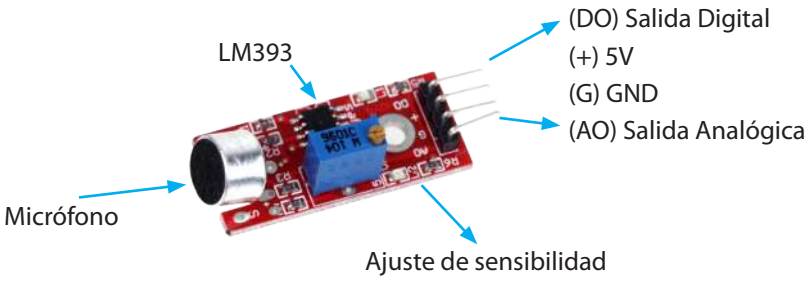


Figura 2. Componentes principales del módulo sensor ky-037.

Fase 2

El circuito que corresponde al semáforo lo puedes revisar en el Taller 3 de este libro. No obstante, siguiendo los siguientes pasos se puede construir rápidamente. Usar la figura 4 como referencia.

1. Disponga ordenados los tres led en la protoboard.
2. Conecte en serie la resistencia de 220Ω al positivo del led rojo, luego el extremo libre de la resistencia conéctelo al pin digital 3 de la placa Arduino.
3. Conecte en serie la resistencia de 220Ω al positivo del led amarillo, luego el extremo libre de la resistencia conéctelo al pin digital 4 de la placa.
4. Conecte en serie la resistencia de 220Ω al positivo del led verde, luego el extremo libre de la resistencia conéctelo al pin digital 5 de la placa.
5. Conecte el negativo de cada led a GND de la placa Arduino.

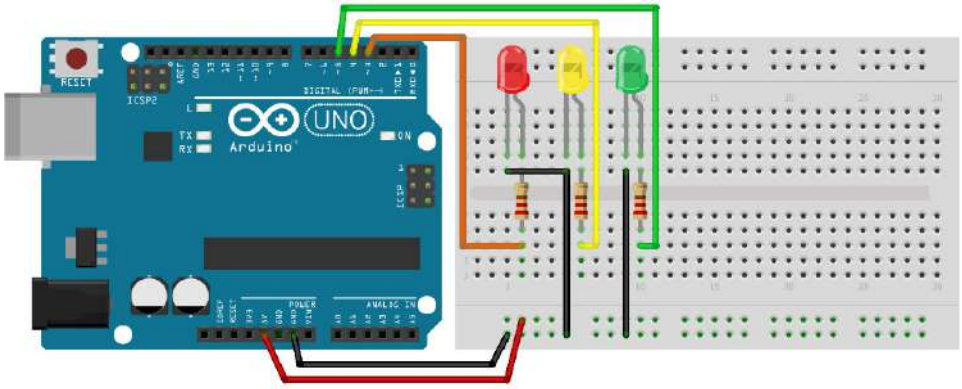


Figura 4. Circuito para el semáforo con LED.

Este semáforo encenderá con una señal de salida alta (HIGH) a través de pin digital respectivo. El estado de la salida de cada pin digital se controla mediante el microcontrolador de la placa Arduino, en respuesta al nivel de sonido detectado por el módulo sensor ky-037. A continuación, se muestra el código ejemplo que permite operar este sonómetro con semáforo.

Código sonómetro con módulo ky-037

```
/* SONÓMETRO SEMÁFORO
   SENSOR KY-037 UTILIZANDO SALIDAS ANALÓGICA Y DIGITAL
   BY LABTEC
*/

// Definiciones
#define PIN_ANALOGO      A0     //SENSOR ANALOGO
#define PIN_DIGITAL      2      //DIGITAL
#define PIN_LED_ROJO     3
#define PIN_LED_AMARILLO  4
#define PIN_LED_VERDE    5

// Variables globales
long veces_leído = 0;
long veces_on = 0;
long Valor = 0;
long rango_led_verde, rango_led_rojo;

// Inicialización
void setup() {
  pinMode(PIN_LED_ROJO, OUTPUT);
  pinMode(PIN_LED_AMARILLO, OUTPUT);
  pinMode(PIN_LED_VERDE, OUTPUT);
  pinMode(PIN_DIGITAL, INPUT);

  digitalWrite(PIN_LED_VERDE, HIGH);
  digitalWrite(PIN_LED_AMARILLO, HIGH);
  digitalWrite(PIN_LED_ROJO, HIGH);
  delay(100);
  digitalWrite(PIN_LED_VERDE, LOW);
  digitalWrite(PIN_LED_AMARILLO, LOW);
  digitalWrite(PIN_LED_ROJO, LOW);

  Serial.begin(9600);
}

// Rutina principal
void loop() {

// Lectura digital
for (veces_leído = 0, veces_on = 0; veces_leído <= 50000; veces_leído++) {
    if (!digitalRead(PIN_DIGITAL)) veces_on++;
   }

// Lectura Analógica
 Valor = analogRead(PIN_ANALOGO);

// Límites del rango de acción de cada led
```

```
  rango_led_rojo  = long (100 + Valor * 10);
  rango_led_verde = long (10 + Valor * 2);

 // Comparaciones y acciones
  if (veces_on <= rango_led_verde) {
    digitalWrite(PIN_LED_ROJO, LOW);
    digitalWrite(PIN_LED_AMARILLO, LOW);
    digitalWrite(PIN_LED_VERDE, HIGH); // verde ON
    delay(10);
  }

  else if (veces_on >= rango_led_rojo) {
    digitalWrite(PIN_LED_ROJO, HIGH); // rojo ON
    digitalWrite(PIN_LED_AMARILLO, LOW);
    digitalWrite(PIN_LED_VERDE, LOW);
    delay(10);
  }
  else {
    digitalWrite(PIN_LED_ROJO, LOW);
    digitalWrite(PIN_LED_AMARILLO, HIGH); // LED amarillo ON
    digitalWrite(PIN_LED_VERDE, LOW);
    delay(10);
  }

 // Monitor serial
  Serial.print(" VECES_ON: ");
  Serial.print(veces_on);
  Serial.print(" VALOR_SENSOR: ");
  Serial.print(Valor);
  Serial.print(" RANGO_ROJO: ");
  Serial.print(rango_led_rojo);
  Serial.print(" RANGO_VERDE: ");
  Serial.println(rango_led_verde);
 }
```

El código fuente está dividido en tres grandes estructuras:

1. Definiciones y variables globales, previo al void setup(): En esta subestructura se definen los pines que se van a utilizar en la placa Arduino, y también son definidas las variables globales del programa.
2. Inicialización, a partir de la ejecución del void setup(): En esta subestructura se configuran las entradas o salidas de la placa Arduino, así como sus estados iniciales.

3. Rutina principal, al ejecutarse el void loop(): En esta subestructura se mide el nivel de sonido a través del sensor usando sus salidas analógica y digital. Se definen los rangos para que encienda el led rojo, verde y amarillo. Y luego se comparan las medidas de cada rango con el número de veces que se supera cierto umbral digital. A partir de esta comparación es posible encender el led verde, amarillo o rojo, según corresponda.

Las principales acciones que se ejecutan en la rutina principal se encuentran descritas a continuación.

Lectura digital

```
for (veces_leído = 0, veces_on = 0; veces_leído <= 50000; veces_leído++) {
   if (!digitalRead(PIN_DIGITAL)) veces_on++;
  }
```

• Primero se lee el valor del PIN DIGITAL del sensor 50.000 veces, y se registran en la variable "veces_on" todas lecturas en que la señal supera el valor umbral.

Lectura Analógica

```
Valor = analogRead(PIN_ANALOGO);
```

• Luego se lee el pin ANÁLOGO del sensor y se asigna a la variable "Valor".

Límites del rango de acción de cada led

```
rango_led_rojo  = long (100 + Valor * 10);
rango_led_verde = long (10 + Valor * 2);
```

• Se establecen los rangos para el accionar de cada led del semáforo. El rango más alto es para encender el led rojo, y el más bajo para encender el led verde. Entre ambos se encuentra el rango para el led amarillo. Nótese que este último rango fue definido implícitamente.

• Hay que tener en cuenta que, al ajustar los rangos de operación de los led, también se están ajustando los niveles de sonido a los cuales se encenderán las luces del semáforo. Así se puede hacer más o menos tolerable al ruido. Es decir, si se define un rango muy bajo para el rojo, ante menores niveles de ruido ambiental se encenderá el led rojo. Algo parecido sucede si se deja muy alto el rango para el led verde: con altos niveles de ruido ambiental, el led verde seguirá encendido. Por lo tanto, al ajustar los rangos, se ajustan los niveles de sonidos en que se encenderán los led del semáforo.

Comparaciones y acciones

```
 if (veces_on <= rango_led_verde) {
   digitalWrite(PIN_LED_ROJO, LOW);
   digitalWrite(PIN_LED_AMARILLO, LOW);
   digitalWrite(PIN_LED_VERDE, HIGH); // LED verde ON
   delay(10);
 }

 else if (veces_on >= rango_led_rojo) {
   digitalWrite(PIN_LED_ROJO, HIGH); // LED rojo ON
   digitalWrite(PIN_LED_AMARILLO, LOW);
   digitalWrite(PIN_LED_VERDE, LOW);
   delay(10);
 }
 else {
   digitalWrite(PIN_LED_ROJO, LOW);
   digitalWrite(PIN_LED_AMARILLO, HIGH); // LED amarillo ON
   digitalWrite(PIN_LED_VERDE, LOW);
   delay(10);
 }
```

• Si la variable "veces_on" es menor o igual que el rango para accionar el led verde, se enciende el led verde.
• Si la variable "veces_on" es mayor o igual que el rango para accionar el led rojo, se enciende el led rojo.
• Si la variable "veces_on" es menor que el rango para accionar el led rojo, pero mayor que el rango para accionar el led verde, se enciende el led amarillo.

Desafíos

1. Calibre el sensor de sonido para que mida el nivel del sonido ambiental al interior de una sala o habitación. Registre el valor de la variable "veces_on" y el valor de los rangos de acción para cada led.
2. Ajuste el semáforo para que indique niveles adecuados de sonido para los usuarios en la sala o habitación donde se desea utilizar.

PROYECTO 2

Construcción y operación de un sensor de CO2 **con indicación de concentraciones y alarma sonora**

En esta actividad se puede construir un sensor que, una vez realizada la calibración, permite medir concentraciones de CO2 en partes por millón (ppm). Con este dato se puede monitorear la calidad del aire que se respira al interior de espacios cerrados, como una sala de clases o una habitación de la casa.

Este sensor permite apreciar las distintas concentraciones del gas medido a través del monitor serial, o bien, en una pantalla LCD. Además, cuando el nivel de concentración del gas supera el umbral tolerable, se activa una señal sonora mediante un pequeño parlante piezoeléctrico. Finalmente, el sensor de gas de este proyecto es de una clase genérica de sensores de la familia MQ, por lo que también sirve de guía para trabajar con muchos otros gases diferentes que puedan ser detectados por los sensores de la familia MQ.

Para la realización de este taller, es necesario disponer de algunos componentes electrónicos y seguir el proceso de construcción que se indica a continuación:

Componentes

- 1 protoboard
- 1 Arduino UNO
- 1 módulo sensor de gas MQ-135
- 1 led rojo.
- 2 resistencias de 220Ω
- 1 buzzer zumbador piezoeléctrico (parlante pequeño)
- Cables para conexiones

Desarrollo

En este proyecto se construirá un sensor de gas con alarma sonora. Para lograrlo se debe armar el circuito que se muestra a continuación en la figura 1. Es importante considerar que el sensor MQ-135 dispone de 4 pines para conexiones. Sin embargo, en este proyecto solo será utilizada la salida analógica (AO) del módulo sensor; es por esta razón que en la figura 1 se muestran solo tres pines: de izquierda a derecha, Vcc, GND y AOUT, respectivamente.

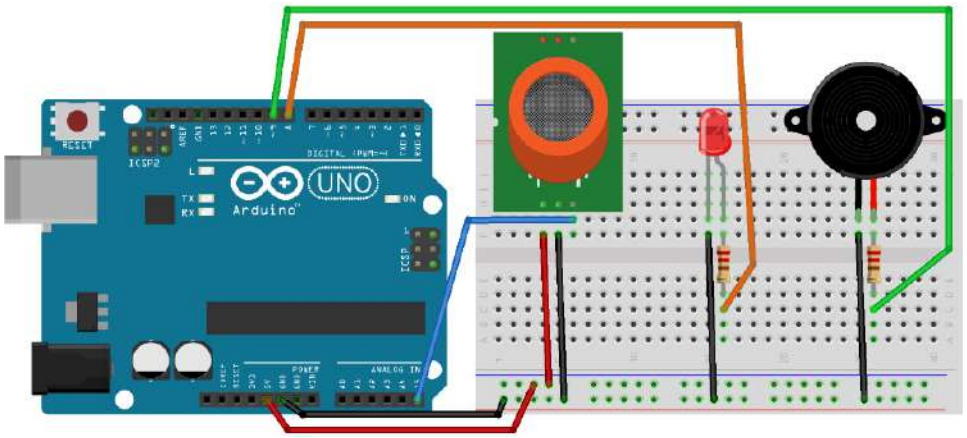


Figura 1. Circuito sensor de gas con alarma sonora.

El sensor MQ-135 consta de 4 pines (ver figura 2): Alimentación 5V (Vcc), tierra (GND), salida analógica (AOUT), salida digital (DOUT). Primero, conecte la salida analógica AO al pin analógico A5 de la placa Arduino. A continuación, relacione la línea de alimentación 5V de Arduino con Vcc del sensor, y finalmente proceda a unir GND de la placa con GND del sensor. Recuerde que en este proyecto solo se utilizará la salida analógica (AOUT) del sensor, y no la salida digital (DOUT).

Figura 2. Pines para conexión eléctrica del sensor MQ-135. Dependiendo del fabricante, puede variar el orden de los pines en el módulo sensor.

El sensor MQ-135 es un sensor de tipo analógico, al igual que todos los sensores de la familia MQ. La detección se basa en un proceso electroquímico, utilizando una resistencia eléctrica que varía con las distintas concentraciones de gas a las que se encuentra expuesto el sensor. Este tipo de sensor trabaja en base a un calentador, por lo que la respuesta depende de que alcance la temperatura de trabajo adecuada. Debido a que este proceso demora entre 12 y 48 horas, para obtener una lectura adecuada es necesario encender el sensor de forma previa a las lecturas de concentración de gas.

En la figura 3 se muestra el circuito esquemático del sensor, junto al circuito equivalente. Se debe tener en cuenta que las concentraciones del gas a medir se realizan de forma indirecta, a partir de la diferencia de potencial Vout que registra la placa Arduino.

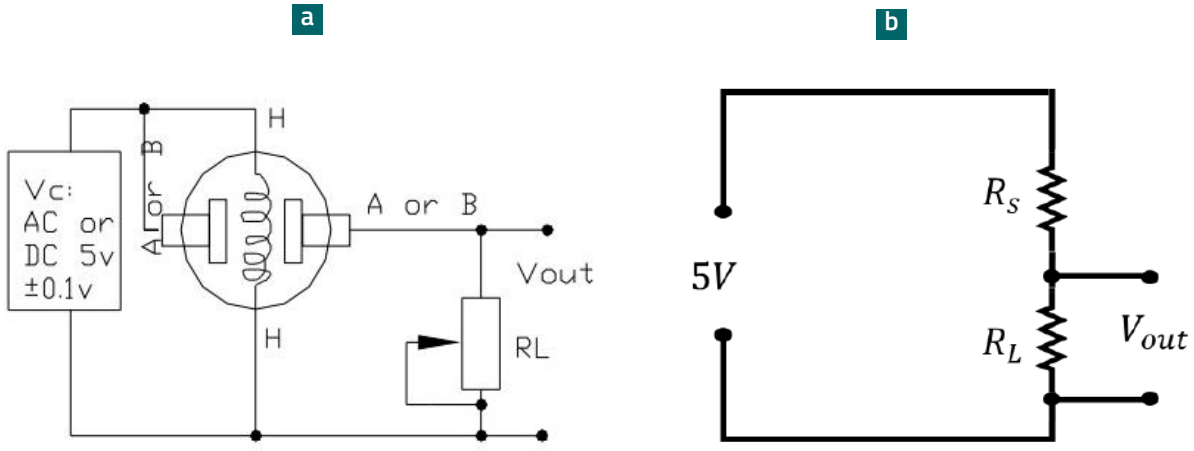


Figura 3. a) Circuito esquemático del sensor MQ-135. b) Esquema circuito equivalente.

Calibración

Con la finalidad de medir las concentraciones de CO2 en el aire al interior de una habitación, se debe tener en cuenta que el sensor debe ser calibrado usando como referencia un sensor estandarizado.

El circuito equivalente del sensor se muestra en la figura 3 b). A partir del análisis del circuito equivalente, se puede demostrar que el valor de la resistencia del sensor se relaciona con los otros parámetros del circuito, a partir de la siguiente expresión:

$$R_s = \frac{R_L(5 - V_{out})}{V_{out}} \qquad (1)$$

Esta última ecuación es la que será modelada en Arduino con tal de determinar las concentraciones del gas de interés. Sin embargo, para conocer las concentraciones de CO2 en alguna media estandarizada, se debe estimar la forma en que R_S varía con dichas concentraciones de gas en el ambiente. Este último dato es entregado por el fabricante del sensor a partir de la hoja de datos[10]. En la figura 4, se puede apreciar cómo varía la resistencia R_S en función de las concentraciones de gas CO2, CO, Alcohol, NH4, Tolueno y acetona. Nótese que el valor de V_{out} se obtendrá a través de una lectura analógica con Arduino, y el parámetro R_L es, por lo general, del orden de 1 kΩ, según se indica en la hoja de datos del sensor.

Puesto que este proyecto tiene por objetivo construir un sensor de CO2, se prestará especial atención a la curva característica que describe el comportamiento del sensor frente a este gas. Para obtener la ecuación que relaciona R_S con las concentraciones del gas en ***ppm***, lo primero es construir una tabla de datos mediante un proceso de regresión, o bien, utilizando algún software para ello. En este caso se recomienda utilizar WebPlotDigitizer[11]. Esta es una herramienta gratuita que permite obtener la tabla de valores deseada a partir del gráfico presente en la hoja de datos del sensor.

Esta tabla de datos deberá ser procesada posteriormente a través de Excel u otro software que nos permita graficar, y obtener de esta forma la ecuación del gráfico respectivo para luego modelarla en Arduino. Es importante destacar que el gráfico entregado por el fabricante indica en el eje vertical la relación R_S/R_O, y en el eje horizontal las concentraciones del gas en ppm. El parámetro R_O debe ser determinado experimentalmente en la fase de calibración del sensor. Téngase en cuenta que R_O corresponde al valor que toma R_S cuando la concentración de CO2 es del orden de unas pocas partes por millón.

10 *https://datasheetspdf.com/pdf-file/605077/Hanwei/MQ135/1*
11 *https://automeris.io/WebPlotDigitizer/*

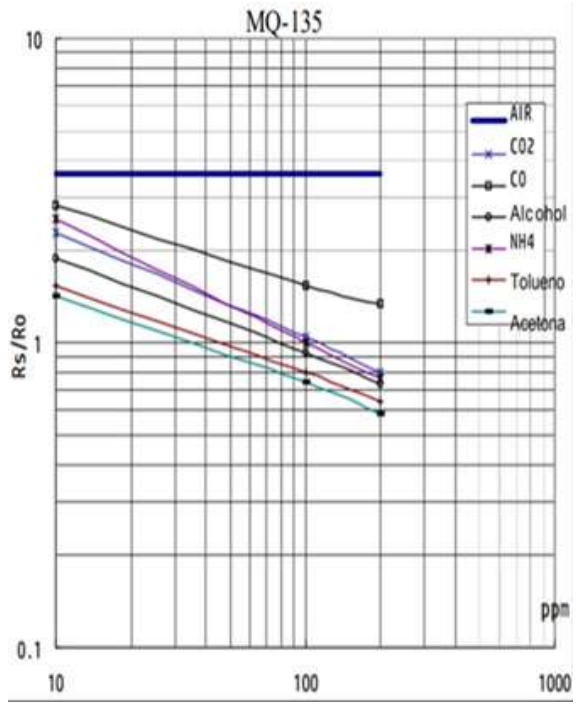


Figura 4. Curva característica del sensor MQ-135 según hoja de datos del fabricante.

Al usar el software WebPlotDigitizer, primero se debe cargar la imagen del gráfico entregado por el fabricante. Luego se escoge un gráfico en dos dimensiones y, con la finalidad de establecer la escala utilizada, se seleccionan 2 puntos de referencia en cada eje, indicando sus respectivos valores. Puesto que de este punto depende que el software establezca la escala de medición adecuada, lo ideal es escoger puntos cuyos valores se lean con claridad en los ejes horizontal y vertical. También es importante que el gráfico se visualice en escala logarítmica en ambos ejes.

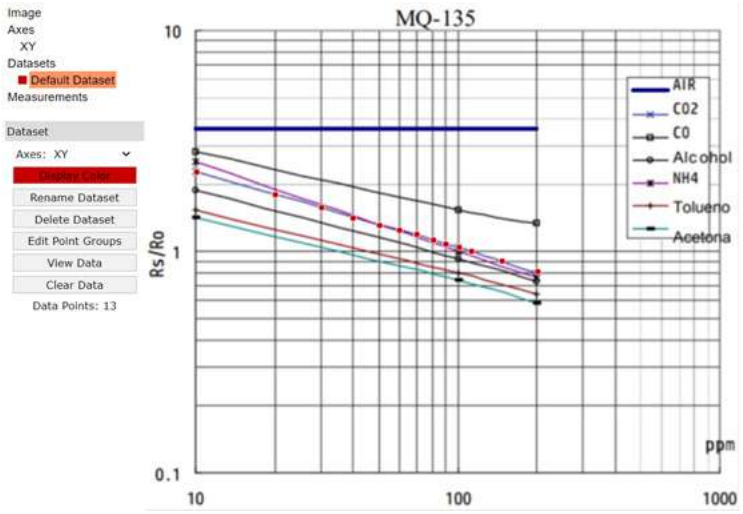


Figura 5. Curva característica del sensor MQ-135 y definición de puntos intermedios usando WebPlotDigitizer.

Establecida la escala de medición, se deben marcar varios puntos intermedios en la curva de interés para obtener la tabla de datos respectiva. Para alcanzar mayor precisión de los valores de esos puntos, se recomienda usar las intersecciones de las rectas verticales y horizontales en el gráfico (ver figura 5). De esta forma, se genera una tabla de valores que se exportan (o descargan) en formato *.CSV para ser utilizada en Excel posteriormente.

PPM de CO2	Rs/Ro
10	2,285463864
19,95988239	1,804721767
29,99806259	1,5768929
39,55090251	1,413133364
49,91858236	1,309833571
59,44175552	1,245197085
69,25359743	1,193776642
80,10017987	1,125335583
89,33607264	1,078864823
100,3644199	1,043073755
111,9368661	1
146,5132079	0,903740939
200,3262017	0,809887896

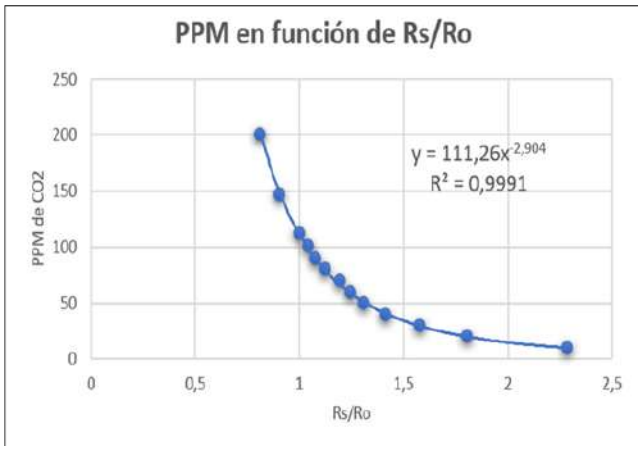


Figura 6. Curva característica del sensor MQ-135 y obtención de ecuación usando Excel.

Ya en la planilla de Excel, se construye el gráfico de concentración de CO2 en partes por millón (ppm) en función de la razón R_s/R_o. Luego se ajusta la línea de tendencia, y se presenta la ecuación del gráfico como se muestra en la figura 6. La ecuación del gráfico tiene la forma general que se indica a continuación:

$$y = a * x^b \quad (2)$$

Donde ***a*** y ***b*** son constantes, ***x*** representa la razón R_s/R_o, e ***y*** representa la concentración de CO2 en ppm. A partir del análisis del gráfico anterior, se obtienen el valor de las constantes ***a*** y ***b***. El valor de R_s se obtiene a partir de la lectura analógica del sensor de CO2 conectado a Arduino. Finalmente, el problema de la calibración se reduce a estimar el valor de R_o. Para ello se sugiere utilizar el siguiente código fuente. Nótese que las constantes y **se** representan como float **A** y float **B** en el programa. En la ecuación (3) se muestra la relación con las otras variables del modelo.

$$ppm = a * (R_s/R_0)^b \quad (3)$$

```
/* SENSOR DE CO2 PARA MEDIR CONCENTRACIÓN EN PPM
   CALIBRACIÓN SENSOR MQ-135
   BY LABTEC
*/

float Rl = 1;  //resistencia incluida en módulo en Kohm
float A = 111.26;
float B = -2.904;

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

void loop () {
  float ADC_MQ135 = analogRead(A5);
  float Vin = (ADC_MQ135 * 5/1024);
  float Rs = Rl * (5 - Vin)/Vin;

//Monitor serial
  Serial.print ("ADC: ");
  Serial.print (ADC_MQ135);
  Serial.print ("  Vin: ");
  Serial.print (Vin);
  Serial.print ("  Rs: ");
  Serial.println (Rs);
  delay(100);
}
```

A partir de este procedimiento, y si todo va bien, se obtendrán las lecturas de R_s a través del monitor serial, como se indica en la figura 7.

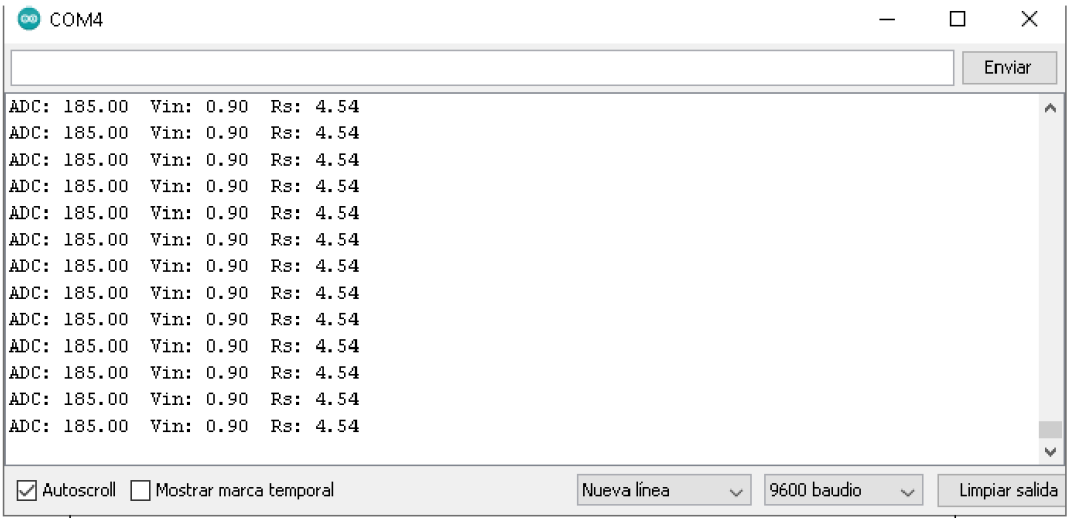


Figura 7. Lecturas obtenidas para el sensor conectado a Arduino.

Los valores de las constantes ***a*** y ***b*** (float A y float B) obtenidos del análisis del gráfico, deben ser reemplazados en el programa de Arduino que modela el comportamiento del sensor.

Con la finalidad de estimar un valor para R_o, se puede usar un sensor de uso comercial (ver figura 8), o algún procedimiento con instrumental de laboratorio más confiable.

En este proyecto, el procedimiento seguido para la obtención de R_o utiliza un sensor de uso comercial. Y los pasos recomendados son los siguientes:

1. Coloque en el mismo lugar el sensor estandarizado de uso comercial y el sensor construido en este proyecto. Por ejemplo, dejarlos sobre una mesa, uno junto al otro.
2. Registre en una tabla de datos, tanto las lecturas de CO2 (en ppm) del sensor estandarizado como las lecturas de R_s entregadas por el sensor conectado a Arduino. Trate de controlar y hacer estables las lecturas de R_s y las concentraciones de CO2 en el ambiente.

Figura 8. Sensor de CO2 de uso comercial utilizado para calibrar el sensor construido en este proyecto.

Concentración de CO2 (en ppm) sensor estándar	R_s lectura dada por el sensor conectado a Arduino	(valor calculado usando ppm y R_s)

Registradas las concentraciones de CO2, en ***ppm***, y el valor de R_s, reemplazar estos valores en la ecuación (4) para obtener el valor aproximado de R_o.

$$R_0 = \frac{R_S}{b\sqrt{\frac{ppm}{a}}} \qquad (4)$$

El valor promedio de la constante R_o debe ser ingresado en el programa "**Código sensor de CO2 MQ-135 para medir concentraciones en ppm**" junto con las constantes A y B. De esta forma se obtendrá -a través del monitor serial de Arduino- las lecturas de las concentraciones de CO2 estandarizadas en ppm.

Código sensor de CO2 MQ-135 para medir concentraciones en ppm

```
/* SENSOR DE CO2 PARA MEDIR CONCENTRACIÓN EN PPM
   SENSOR MQ-135
   BY LABTEC
*/

#define ledPin 8
#define piezoPin 9

float Rl = 1;  //resistencia incluida en módulo en Kohm
float Ro = 8.20; //reemplazar por el valor obtenido en la calibración
float A = 111.914; //ingresar dato obtenido desde el análisis del gráfico
float B = -2.871; //ingresar dato obtenido desde el análisis del gráfico

byte pinState = 0; //LED control
int count = 0;
int x = 0;

void setup () {
  Serial.begin (9600);
  pinMode(piezoPin, OUTPUT);
  beep(50);
  beep(50);
  beep(50);
  pinMode(ledPin, OUTPUT);
  delay(1000);
}

void loop () {
  toggle(ledPin);

//MQ135
  float ADC_MQ135 = analogRead(A5);
  float Vin = (ADC_MQ135 * 5/1024);
  float Rs = Rl * (5 - Vin)/Vin;
  float CO2 = A * pow((Rs/Ro), B);

  if (CO2 >= 5000) {
    for(x = 0; x < 10; x++){
      beep(50);
```

```
   toggle(ledPin);
    }
 }

//Monitor serial
  Serial.print ("ADC: ");
  Serial.print (ADC_MQ135);
  Serial.print ("  Vin: ");
  Serial.print (Vin);
  Serial.print ("  Rs: ");
  Serial.print (Rs);
  Serial.print ("  CO2: ");
  Serial.print (CO2);
  Serial.println (" ppm");
  delay(100);
}

//=== Subrutinas ===
// Beep piezo speaker
void beep(unsigned char delayms) {
  analogWrite(piezoPin, 20);
  delay(delayms);
  analogWrite(piezoPin, 0);
  delay(delayms);
}

// LED control to Blink
void toggle(int pinNum) {
   digitalWrite(pinNum, pinState);
   pinState = !pinState;
  //delay(500);
}
```

Desafíos

1. Incluir una pantalla LCD de 16x2 que muestre las lecturas de las concentraciones de CO2.
2. Establezca otro procedimiento para calibrar las lecturas del sensor de CO2 en ppm. Consulte con su profesora o profesor de química para ello.

PROYECTO 3

Construcción de un robot impreso en 3D y **navegación autónoma usando sensor de distancia por ultrasonido**

En esta actividad se puede construir un robot que navegue usando dos servomotores de giro continuo y un sensor de distancia por ultrasonido. Para la construcción del chasis se puede usar distintos materiales. No obstante, los archivos que permiten construirlo usando una impresora 3D, están disponibles en el siguiente link https://bit.ly/ImPreSion3D. Este chasis les permitirá construir un robot que puede ser utilizado en muchas otras tareas que impliquen navegar en un espacio plano, usando diversos sensores e incluso combinándolos.

Para la realización de este taller, es necesario disponer de algunos componentes electrónicos y seguir el proceso de construcción que se indica a continuación:

Componentes

- 1 Arduino UNO
- 1 shield para prototipos para Arduino UNO con mini protoboard
- 1 sensor de distancia HC-SR04
- 2 servomotores estándar de giro continuo o 360°
- 1 portapilas para 4 pilas en serie (6V)
- 4 pilas AA
- Cables para conexiones
- Chasis para el robot (lo puedes descargar desde https://bit.ly/ImPreSion3D)
- 8 pernos cocina con tuerca 1/8"x1/2"
- 1 bolita de cristal o esfera de 15mm de diámetro

Actividades previas relacionadas

- **Taller 8:** Medición de distancia con sensor de ultrasonido
- **Taller 9:** Servomotores y rutinas de movimiento

Desarrollo

En este proyecto se construirá un robot que puede navegar en un espacio plano, como la cubierta de una mesa o el piso de una sala. Para lograrlo se debe armar el robot que se muestra en la figura 1, y luego programarlo con el código fuente de ejemplo que se indica en este mismo proyecto.

El que aquí se presenta será desarrollado en tres fases:

I) Impresión 3D
II) Armado del robot
III) Programación para la navegación autónoma usando sensor de distancia HC-SR04.

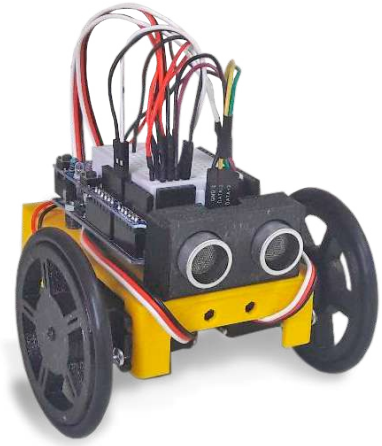


Figura 1. Robot LabTec armado completamente.

Fase 1: Impresión 3D

Imprima las partes componentes del robot usando una impresora 3D. El chasis del robot está compuesto de tres partes principales que se muestran en la figura 2. Estas partes son: i) el cuerpo principal, ii) las ruedas, y iii) y la caja soporte para el sensor de ultrasonido.

Figura 7. Lecturas obtenidas para el sensor conectado a Arduino.

El cuerpo principal dispone de un espacio para colocar un portapilas específico. No obstante, se puede utilizar con cualquier batería de 6V, realizando los ajustes necesarios para ello. En la figura 3 se muestran las partes que componen el cuerpo principal del chasis.

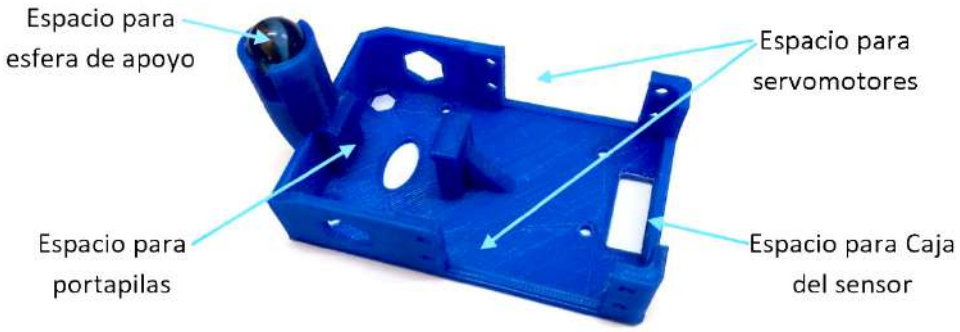


Figura 3. Cuerpo principal del chasis con detalle de sus partes (vista inferior).

Fase 2: Armado del robot

Para el armado del robot se deben seguir las instrucciones que se detallan a continuación.

Paso 1: Monte la placa Arduino sobre el cuerpo principal del robot. Se sugiere usar unos pequeños espaciadores para que la placa quede unos 5mm separada del cuerpo principal. Fíjela usando los pernos necesarios. Una vez que quede fija la placa Arduino, no olvide montar el Shield para prototipos sobre la placa. En la figura 4 se muestran los componentes utilizados en este paso.

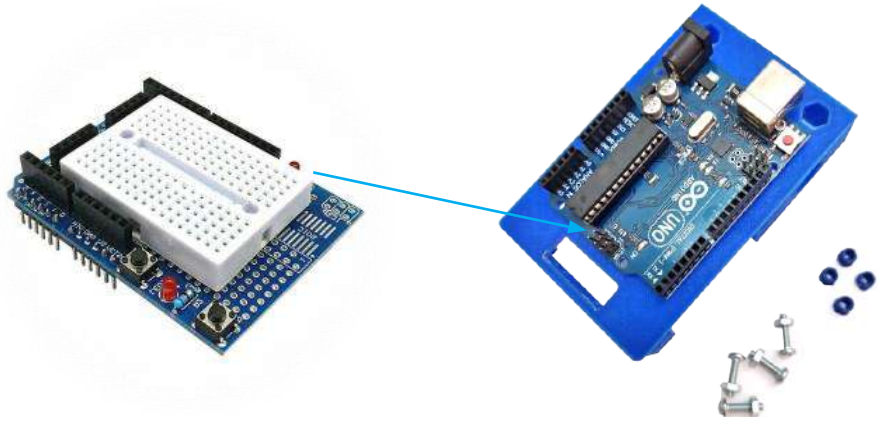


Figura 4. Paso 1: Montaje de plana Arduino UNO al cuerpo principal del chasis (vista superior), junto al Shield para prototipos.

Paso 2: Coloque los servomotores en los espacios respectivos. Procure que el rotor del servomotor (donde se colocan las ruedas) quede desplazado hacia el frente del robot. Fíjelos usando al menos dos pernos, e inserte la esfera de apoyo (esfera de cristal en la figura 5) en el lugar que corresponde. En la figura 5 se muestra la posición adecuada para los servomotores y los pernos de fijación.

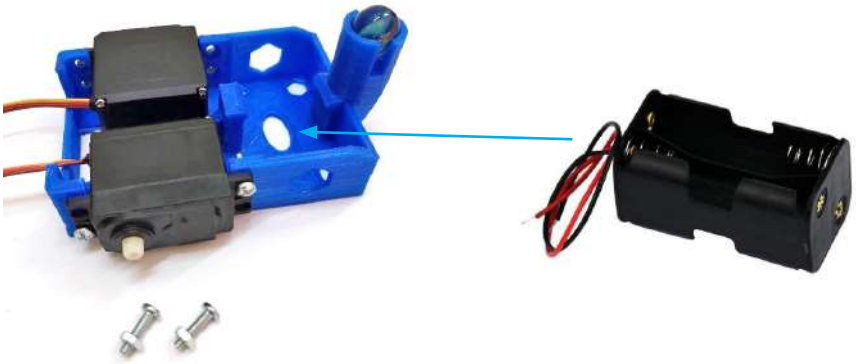


Figura 5. Paso 2: Montaje de plana Arduino UNO al cuerpo principal del chasis (vista superior), junto al portapilas.

Paso 3: Coloque el sensor de distancia en la caja soporte. Luego, monte la caja en el espacio respectivo del cuerpo principal. Nótese que los pines para las conexiones de cables del sensor deben quedar hacia arriba.

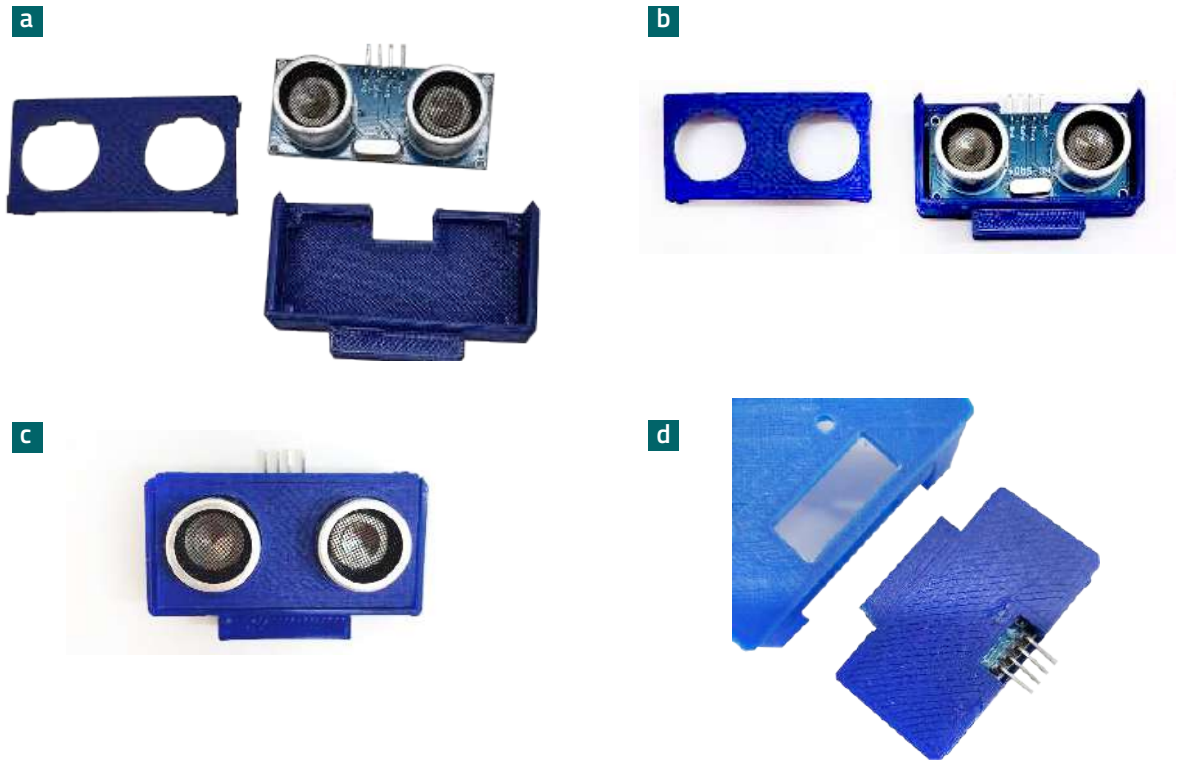
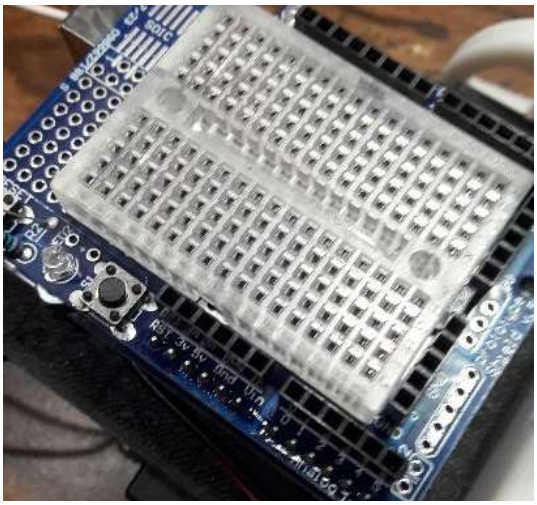


Figura 6. Paso 3: Montaje del sensor de distancia en el cuerpo principal del chasis.

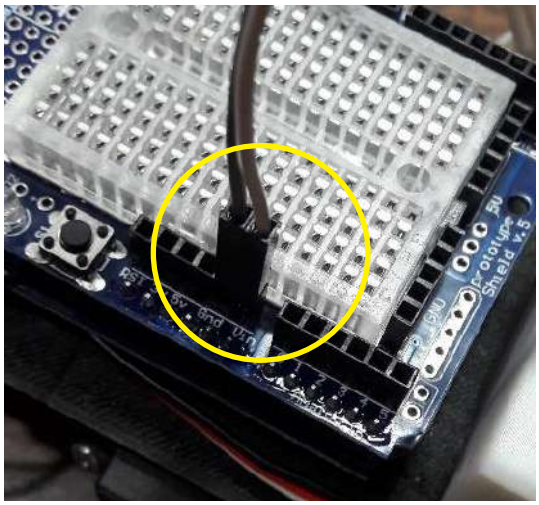
Paso 4: Realizar las conexiones eléctricas del robot. Para ello debe prestar atención a secuencia que se indica a continuación.

Conexión de los servomotores

a Vista superior del Shield para prototipos.

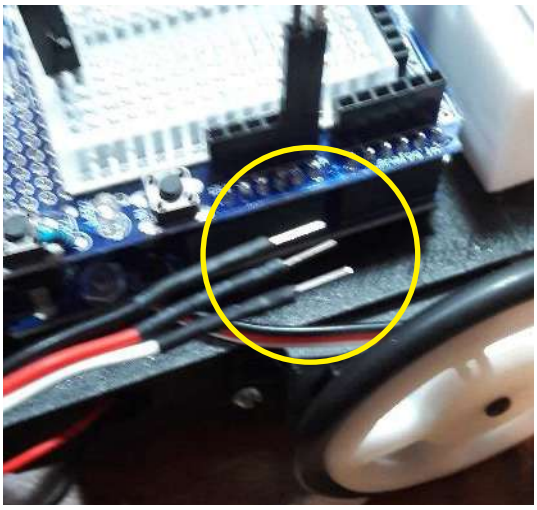


b Conecte dos claves: uno al GND, el otro al Vin del Shield.

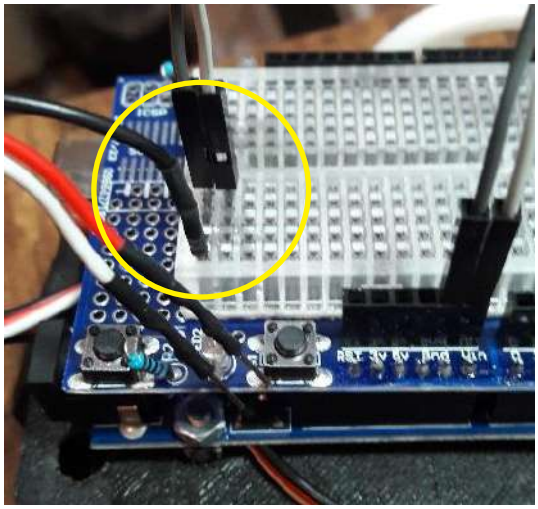


c Conecte los extremos libres de los cables en la protoboard.

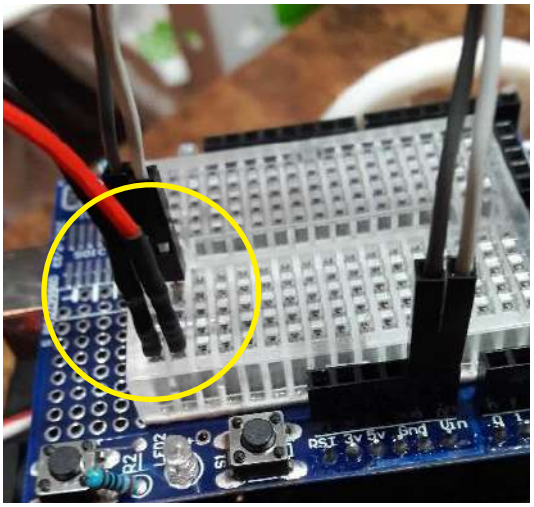
d Cada servomotor dispone de tres cables: GND, Vcc, control.



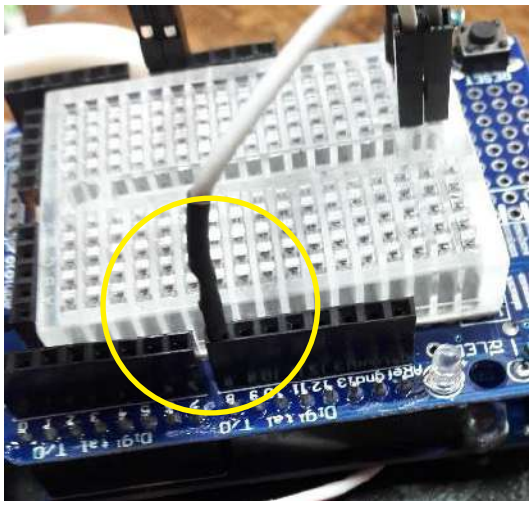
e Conecte el cable GND del servomotor en la línea GND de la protoboard.



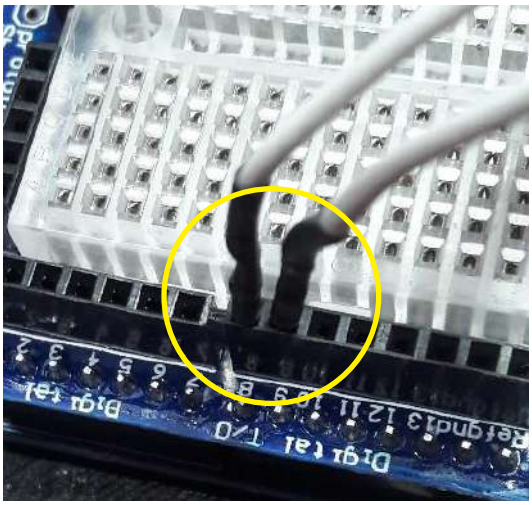
f Luego, conecte el cable Vcc del servomotor a la línea Vin de la protoboard.



g Conecte el cable de control del servo al pin8 del Shield.

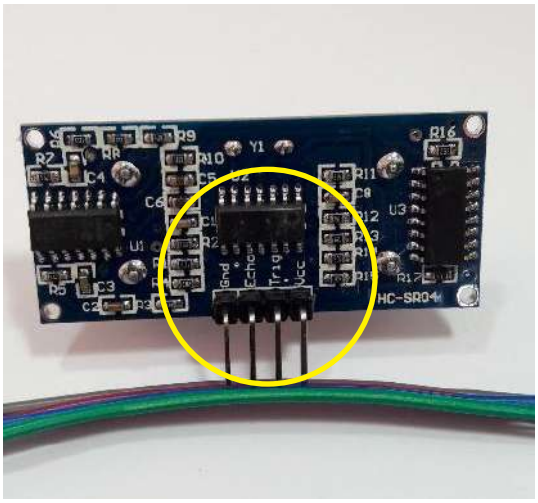


h Repita los pasos e) y f) con el otro servomotor y conéctelo al pin 9 del Shield

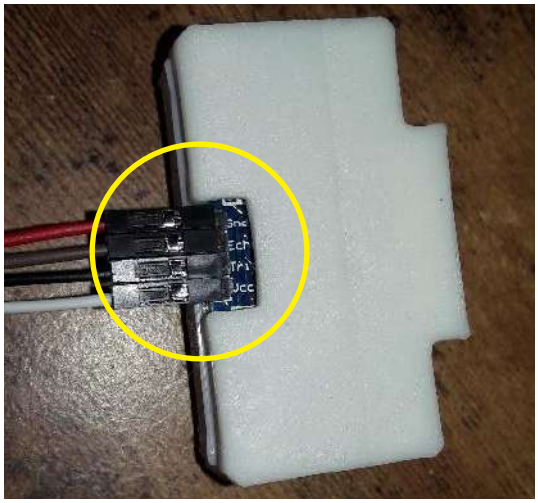


Conexión del sensor de distancia HC-SR04

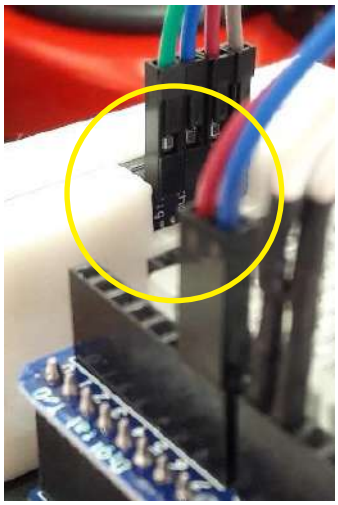
i El sensor HC-SR04 dispone de 4 pines: GND, Echo, Trigger, Vcc.



j Conecte un cable a cada pin del sensor HC-SR04



k Conecte el cable Trigger y Echo del sensor a los pines 6 y 7 del Shield.



l Conecte el cable GND del sensor al GND del Shield, y el Vcc del sensor a los 5V del Shield.

Finalmente, recuerde conectar la batería de 6V (el portapilas con 4 pilas AA en serie) a la protoboard del Shield. Las conexiones deben realizarse de forma que el negativo de la batería vaya a la línea GND y el positivo de la batería se vincule con la línea Vin, en las mismas líneas donde se conectaron el GND y el Vcc de los servomotores, sobre la protoboard del Shield (ver imagen c) en la conexión de los servomotores).

Código para la navegación autónoma usando el sensor de distancia

```
/* Medición de distancia con HC-SR04
 * Navegación autónoma
 * by LabTec
 */

#include <Servo.h>

// Creación de objetos y variables globales
Servo servoI, servoD;

int x = 0, a;
int der = 0, izq = 0;
const int Mder = 8;
const int Mizq = 9;
long duration, cm = 0, cm1, cm2;
const int echoPin = 5;  //input
const int trigPin = 4;  //output
const int dist = 35;   //distancia límite
const int pausa = 10;   //pausa general

// Inicialización
void setup() {
  servoI.attach(Mizq);
  servoD.attach(Mder);
  servoD.write(90);
  servoI.write(90);
  delay(2000);
}

// Rutina principal
void loop() {
// mide distancia frontal y detecta obstáculos
  cm = sonar();

//si la distancia al obstáculo es menor que la distancia limite
 if (cm <= dist) {
//gira ligeramente hacia un lado y mide distancia 1
    for (x = 0; x < 30; x++) {
      servoD.write(80);
      servoI.write(80);
      delay(pausa);
      }
    cm1 = sonar();
//gira ligeramente hacia el otro lado y mide distancia 2
    for (x = 0; x < 60; x++) {
      servoD.write(100);
      servoI.write(100);
      delay(pausa);
}
```

```
   cm2 = sonar();
//compara distancias y actúa de acuerdo a dónde hay más espacio libre
if (cm2 < cm1) {
     a = random(40, 120);    //variable aleatoria para salir de atascos
     if (der > 2) {
       der = 0;
       for (x = 0; x < a; x++) {
         servoD.write(70);
         servoI.write(70);
         delay(pausa);
       }
     }
     else {
       for (x = 0; x < 45; x++) {
         servoD.write(70);
         servoI.write(70);
         delay(pausa);
       }
       der++;
     }
   }
//giro
if (cm2 > cm1) {
     a = random(40, 120);
     if (izq > 2) {
       izq = 0;
       for (x = 0; x < a; x++) {
         servoD.write(110);
         servoI.write(110);
         delay(pausa);
       }
     }
     else {
       for (x = 0; x < 45; x++) {
         servoD.write(110);
         servoI.write(110);
         delay(pausa);
       }
     }
     izq++;
   }
 }
//avance
 servoD.write(110);
 servoI.write(70);
 delay(pausa);
}

// Subrutinas
long sonar()
```

```
{
  pinMode(trigPin, OUTPUT);
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(trigPin, LOW);

  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  cm = duration / 29 / 2;
  return cm;
}
```

El código fuente está dividido en cuatro grandes estructuras:

- Creación de objetos y variables globales, previo al void setup(): En esta subestructura se definen los pines que se van a utilizar en la placa Arduino, y también son definidas las variables globales del programa.
- Inicialización, a partir de la ejecución del void setup(): En esta subestructura se configuran las entradas o salidas de la placa Arduino, así como sus estados iniciales.
- Rutina principal, al ejecutarse el void loop(): En esta subestructura se observa si hay obstáculos frente al robot, se mide la distancia respecto a dichos obstáculos y se toman decisiones para la navegación autónoma del sistema.
- Subrutinas, donde se estima la distancia mediante el sensor HC-SR04. Esta medición se realiza a partir de un pulso de ultrasonido que viaja por el aire y se refleja en los obstáculos que encuentra enfrente.

Desafíos

1. Ajuste la distancia límite de detección del robot y permita que navegue de forma autónoma sobre un espacio plano al interior de una habitación. Ajuste lo necesario para que no quede atascado en algún obstáculo.
2. Construya un laberinto de obstáculos y ajuste el programa que gobierna el robot para que pueda salir de él. Este laberinto lo puede construir con cajas de cartón, plástico o cualquier material que refleje los pulsos de sonido que salen desde el sensor.

PROYECTO 4

Regla Digital con Bluetooth

En esta actividad se construirá una regla digital de tamaño reducido, capaz de medir la distancia de objetos desde 2 cm hasta cerca de 4 metros, mostrar las mediciones a través de una pantalla LCD y enviar los datos a otro dispositivo vía bluetooth (opcional).

Para la realización de este taller, es necesario disponer de algunos componentes electrónicos y seguir el proceso de construcción que se indica a continuación:

Componentes

- 1 protoboard
- 1 Arduino UNO
- 1 display LCD I2C
- 1 módulo bluetooth HC-06
- 1 módulo sensor ultrasónico HC-SR04
- Cables para conexiones

Actividades previas relacionadas

- Taller 8: Medición de distancia por ultrasonido
- Taller 10: Pantalla LCD de 16x2
- Taller 12: Comunicación por Bluetooth (opcional)

Desarrollo

En este proyecto de la regla digital indicadora de distancia, deberás armar el circuito que se muestra a continuación en la figura 1.

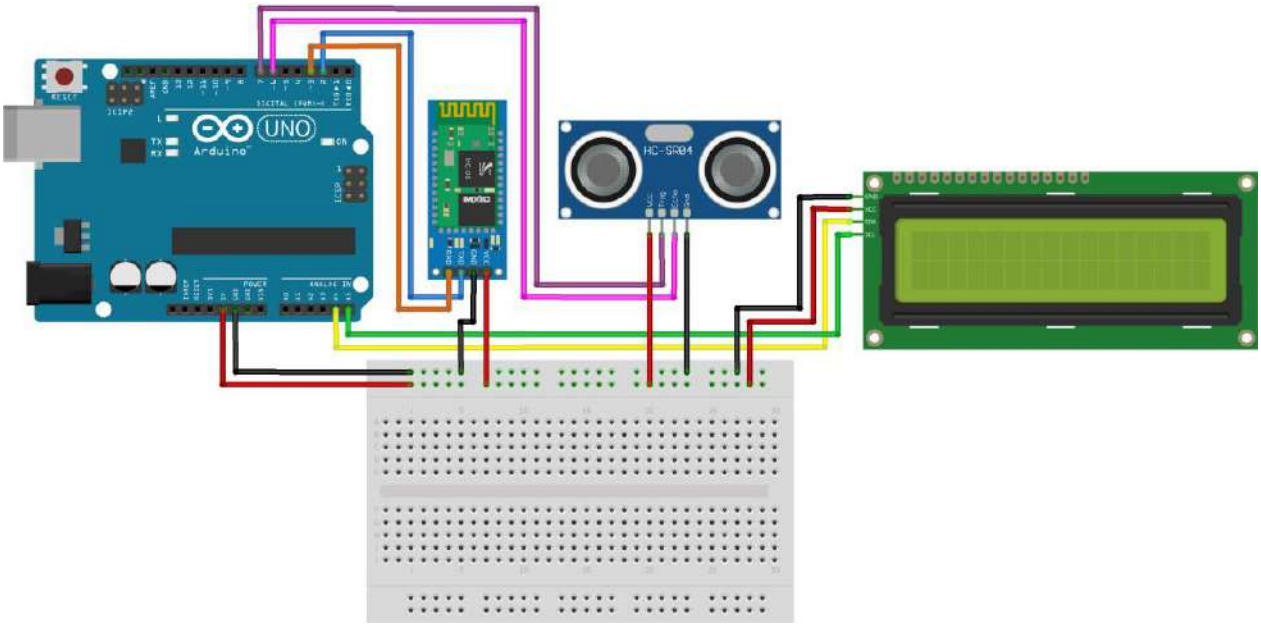


Figura 1. Esquema del circuito de regla digital.

Para construir el circuito final debe seguir las instrucciones, que dividiremos en tres fases:

Fase 1

Primero, conecte al protoboard el módulo sensor ultrasónico HC-SR04, luego conecte a la placa Arduino, como muestra la Figura 1. El sensor ultrasónico HC-SR04 genera una onda sónica en el emisor mediante un pulso que pone "trig" (trigger o disparador en pin digital 7 de la placa). Esta onda se reflejará al encontrarse con algún obstáculo, volviendo de esa manera al sensor para ser registrada por el receptor, el cual traduce esta onda reflejada llamada "Echo" (pin digital 6 de la placa).

A continuación, conecte la línea de alimentación a 5V de Arduino con VCC (+) del sensor. Finalmente, conecte GND de la placa con GND del sensor. En la figura 4 se muestran las partes principales del módulo.

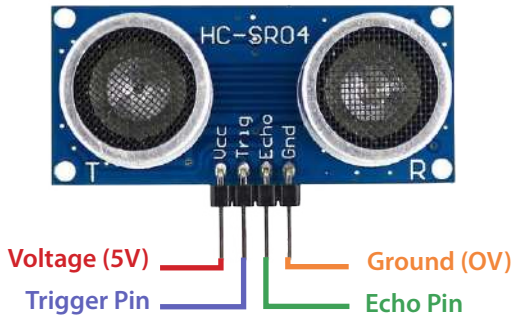


Figura 4. Componentes principales del módulo sensor ultrasónico HC-SR04.

Fase 2

Ahora, ubique la pantalla LCD -que cuenta con un módulo I2C en la protoboard- y conéctelo a la placa Arduino (ver Figura 1). El display LCD se conecta a los pines I2X del Arduino (SDA análogo 4 y SCL análogo 5). alimentado en VCC (5V) y conectado a GND (tierra). Para el funcionamiento de este módulo se utilizan las librerías LiquidCrystal.h y Wire.h de Arduino (en la Figura 3 se muestran las partes principales de la pantalla LCD).

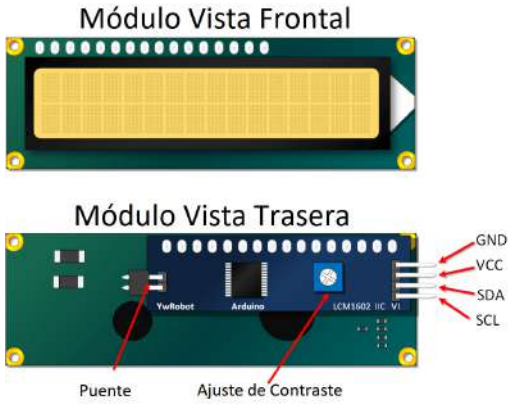


Figura 3. Componentes principales de display LCD con I2C.

Si bien ya es posible utilizar la regla digital, se propone la tercera fase con la inclusión de un módulo bluetooth para Arduino a la regla digital, con la cual usted puede obtener las medidas desde cualquier dispositivo.

Fase 3

Se utilizará el módulo bluetooth HC-06. Ubíquelo en el protoboard y conéctelo a la placa Arduino (ver Figura 1). El circuito que corresponde al módulo bluetooth HC-06 se conecta a TXD, que es el pin de transmisión de datos: con este pin, el módulo HC-06 transmite los datos desde la aplicación móvil "Bluetooth Terminal". Este pin se conectará con la placa Arduino al pin digital 2; luego, RXD es el pin de recepción a través del cual el HC-06 recibirá los datos del Arduino, los cuales se transmiten por bluetooth. Este pin va conectado al pin digital 3 de la placa.

A continuación, conecte la línea de alimentación a 5V de Arduino con VCC (+) del sensor, y finalmente conecte GND de la placa con GND del sensor (en la Figura 2 se muestran las partes principales del módulo). Por último, para administrar el módulo en el programa, la biblioteca utilizada es SoftwareSerial.h que administra los enlaces seriales de la placa Arduino.

A continuación, se muestra el código ejemplo que permite operar la regla digital.

Código Regla Digital con Módulo HC-SR04

```
/* REGLA DIGITAL
  SENSOR HC-SR04 UTILIZANDO SALIDAS DIGITALES
/*

//Librerías
#Include <Softwareserial.H>
#Include <Liquidcrystal_I2c.H>
#Include<Wire.H>

//Variables
Softwareserial Bt(4,2);
Liquidcrystal_I2c Lcd(0x3f,16,2);
Long Distancia;
Long Tiempo;

// Inicialización
Void Setup() {
  Lcd.Init();
  Bt.Begin(9600);
  Lcd.Backlight();
  Serial.Begin(9600);
  Pinmode(2, Output);
  Pinmode(4, Output);
  Digitalwrite(4, High);
  Delay (2000) ;
  Serial.Println("Inicializando Módulo HC-06");
  Digitalwrite (2, High);
  Bt.Begin(9600);
  Pinmode(9, Output);
  Pinmode(8, Input); }

// Rutina principal
 Void Loop() {
```

```
// Comparaciones y acciones
  Digitalwrite(9,Low);
  Delaymicroseconds(5);
  Digitalwrite(9, High);
  Delaymicroseconds(10);
  Tiempo=Pulsein(8, High);
  Distancia= Int(0.017*Tiempo);




// Monitor Serial
  Lcd.Setcursor(0,0);
  Lcd.Print("Distancia ");
  Lcd.Setcursor(0,1);
  Lcd.Print(Distancia);
  Lcd.Setcursor(5,1);
  Lcd.Print("cm");
  Delay(2000);
  Lcd.Clear();
  Bt.Println(Distancia);
  Bt.Println("cm");
  If(Bt.Available())
  Serial.Write(Bt.Read());
  Bt.Println(Distancia);
   If (Serial.Available())
  Bt.Write(Serial.Read());  {
    Serial.Write(Bt.Read());
    Lcd.Write (Serial.Read ());
     If (Bt.Available())
          Serial.Write(Bt.Read());
     If (Serial.Available())
        Bt.Write(Serial.Read()); }
 }
```

Desafíos

1. Utilice una regla o huincha de medir para verificar que concuerden los datos registrados en la aplicación móvil. ¿Cuáles es el valor más pequeño y el valor máximo que puede medir con la regla digital?

2. ¿Será posible medir cuerpos en oscilación, como un péndulo o un resorte? De ser posible, ¿qué rol juega la frecuencia de mediciones?

PROYECTO 5

Microestación de monitoreo

En esta actividad se puede construir una microestación de monitoreo ambiental autónoma. El diseño propuesto será capaz de monitorear temperatura, humedad relativa del aire e intensidad de luz, valores que podrán ser posibles de visualizar por medio de una pantalla LCD. La autonomía estará dada por la batería externa que conecte a la microestación, permitiendo medir en espacios abiertos con duración prolongada.

Para la realización de este proyecto, es necesario disponer de algunos componentes electrónicos y seguir el proceso de construcción que se indica a continuación:

Componentes

- 1 protoboard
- 1 Arduino UNO
- 1 resistencias 5 kΩ
- 1 resistencias 10 kΩ
- 1 display LCD I2C
- 1 módulo sensor de temperatura y humedad DHT11
- 1 módulo sensor fotorresistor LDR
- Cables para conexiones

Actividades previas relacionadas

- **Taller 4**: Humedad y temperatura ambiental
- **Taller 10**: Pantalla LCD 16x2
- **Taller 15**: Sensor de luz analógico

Desarrollo

En este proyecto de la microestación, debe armar el circuito que se muestra a continuación en la Figura 1.

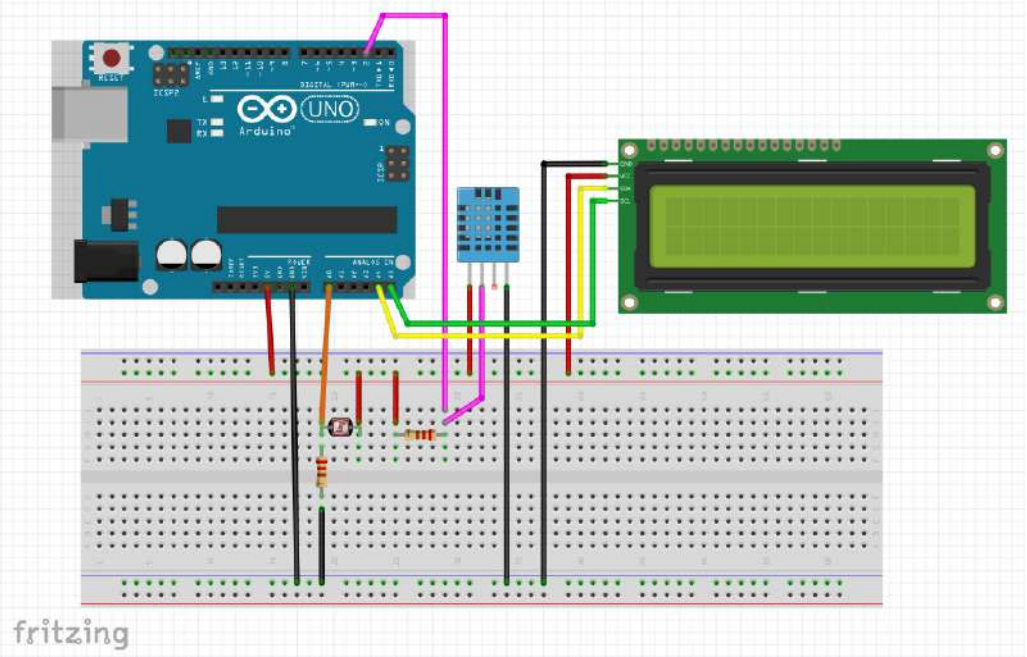


Figura 1. Esquema de circuito de microestación de monitoreo ambiental.

Para construir el circuito, se propone seguir las instrucciones que hemos dividido en las siguientes 4 fases:

Fase 1
Ubique el módulo de temperatura y humedad relativa DHT11 en la protoboard y conéctelo a la placa Arduino. Usar como referencia la Figura 1.

El circuito que corresponde al módulo DHT11 necesitará tener una resistencia de 5 kΩ conectada a la salida digital I/O; la salida NC no estará conectada. A continuación, conecte la línea de alimentación a 5V de Arduino con VCC (+) del sensor, y finalmente conecte GND de la placa con GND del sensor. Por último, para administrar el módulo en el programa, la biblioteca utilizada es Adafruit Unified Sensor Driver que se ocupa para obtener la información de temperatura y humedad.

Fase 2
Ubique el módulo Sensor Fotorresistor LDR en la protoboard y conéctelo a la placa Arduino. Usar como referencia la Figura 1.

En el circuito que corresponde el Fotorresistor LDR, una de las patillas se conecta a 5V, otra patilla con una resistencia de 10 kΩ, mientras que la otra pata de la resistencia se conecta a una entrada analógica (A0) en la placa del Arduino. Un Fotorresistor disminuye su resistencia a medida que aumenta la luz sobre él (en la Figura 2 se muestran las partes principales del módulo).

Figura 2. Componentes principales del módulo sensor fotorresistor LDR.

Fase 3
Coloque el display LCD I2C en la protoboard y conéctelo a la placa Arduino. Usar como referencia la Figura 1.

El circuito que corresponde al display LCD I2C se conecta con los pines I2X del Arduino (SDA análogo 4 y SCL análogo 5) y es alimentado por GND (tierra) y VCC (5V). Por último, se utiliza la librería LiquidCrystal y Wire.h de Arduino (en la figura 4 se muestran las partes principales del módulo).

Fase 4
La última fase consiste en realizar una caja capaz de cubrir el circuito de la microestación, con la finalidad de protegerlo del polvo y humedad, además de posibilitar el transporte de la microestación.

A continuación, se muestra el código ejemplo que permite operar la microestación de monitoreo ambiental.

Código Microestación de monitoreo ambiental con módulo DHT11 y fotorresistor LDR

```
/* Microestación de monitoreo ambiental
  SENSOR DHT11 UTILIZANDO SALIDA DIGITAL
  SENSOR FOTORRESISTOR LDR UTILIZANDO SALIDA ANÁLOGA
/*

//Librerías
#include <Adafruit_Sensor.h>
#include <DHT.h> //librería para controlar el sensor DHT11
#include <LiquidCrystal_I2C.h> //librería para controlar la pantalla LCD

// Se define el pin digital donde se conecta el sensor, en este caso será el pin 2
#define DHTPIN 2
#include <Wire.h>

// Dependiendo del tipo de sensor será DHT11 o DHT22, en este caso se propone el DHT11
#define DHTTYPE DHT11

// Se inicializa el sensor DHT11, ya declarado en DHTTYPE, en el pin 2 declarado en DHTPIN
DHT dht(DHTPIN, DHTTYPE);

//Se define el espacio de trabajo de la pantalla
LiquidCrystal_I2C lcd(0x3F,16,2);

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 (fotosensor)
int V; //variable que almacenara el voltaje obtenido
int ilum; //variable que almacenara la luminosidad

void setup() {
// Se debe inicializar la comunicación en serie
Serial.begin(9600);
// Comenzamos el sensor DHT
dht.begin();
pinMode(4, OUTPUT); // alimentación DHT11
}

// Rutina principal
void loop() {

// Comparaciones y acciones
delay(2500);
digitalWrite(4, HIGH);

// Se definen las variables que leerán humedad relativa (hum) y temperatura (temp)
float hum = dht.readHumidity();
// Leemos la temperatura en grados centígrados (por defecto)
float temp = dht.readTemperature();
```

```
//*****************************************************************

V = analogRead(LDRPin); //realiza medición del voltaje para luego calcular la luminosidad
//ilum = ((long)(1024-V)*A*10)/((long)B*Rc*V); //usar si LDR se encuentra entre GND y A0
ilum = ((long)V*A*10)/((long)B*Rc*(1024-V)); //usar si LDR se encuentra entre A0 y Vcc (como en el esquema anterior)

// *****************************************************************

// Monitor serial
lcd.init(); // Inicializa display
lcd.backlight(); // Inicia luz del display

lcd.setCursor(0,0);
lcd.print("Microestacion");
lcd.setCursor(0,1);
lcd.print("  de Monitoreo");


// ************************************************

//if(hum < 50){
// digitalWrite(8, LOW);
//}
//if(hum > 50){
// digitalWrite(8, HIGH);
//}
// ************************************************

delay(2000); //espera 2 segundos

lcd.clear();
lcd.setCursor(0,0); // Sitúa el cursor en la columna 0, en la fila 0.
lcd.print("Humedad: "); // Todo lo que se escriba entre comillas es textual en el display
lcd.print(hum); //imprime el valor de la variable hum
lcd.print("%");
lcd.setCursor(0,1); // Sitúa el cursor en la columna 0, en la fila 1.
lcd.print("Temp: ");
lcd.print(temp); //imprime el valor de la variable temp
lcd.print("C");

delay(2000); //espera 2 segundos

lcd.clear();
lcd.setCursor(0,0);
lcd.print("Luminosidad: ");
lcd.setCursor(0,1);
lcd.print(ilum); //imprime el valor de la variable ilum
}
```

Desafío

Realice una comparación de luminosidad en el interior de la sala y fuera del aula.2) Explore una relación funcional entre los parámetros ambientales que pueda medir, tanto al interior como fuera del aula.

PROYECTO 6

Huerto con **riego automatizado**[12]

12 *Sepúlveda N. Huertos Inteligentes con Arduinos, Propuesta de Contextualización de Cátedra de Electricidad y Magnetismo para Ingeniería. SOCHEDI. 2019.*

Alerta: Este proyecto involucra condiciones de exposición a peligro de electrocución, por lo cual debe ser asesorado necesariamente por un adulto con extrema precaución.

Este proyecto consiste en el diseño y construcción de un sistema de regadío automatizado, automatización que depende directamente de las condiciones de humedad que presente el suelo. Para ello se implementa el sensor de humedad YL-69.

Es necesario que se construya el lugar para el huerto, se diseñe el camino que tomará el agua y que se implemente por medio de mangueras, junto con escoger una bomba de agua adecuada para el proyecto. Las bombas necesitan una cantidad de corriente mayor, por lo cual se necesita un dispositivo electromagnético llamado Relé.

Componentes

- 1 protoboard
- 1 Arduino UNO
- Cables para conexiones
- 1 relé
- Sensor YL-69
- Bomba para agua (tipos de bomba, fotografías y características de las bombas)
- Contenedor para el agua (Bidón 5 Lt)
- Mangueras
- Caja o contenedor huerto
- Pantalla LCD (opcional)

Actividades previas relacionadas

- **Taller 4:** Humedad y temperatura ambiental
- **Taller 5:** Humedad del suelo
- **Taller 10:** Pantalla LCD 16x2 (opcional)

Desarrollo

Este proyecto está diseñado en 4 fases para que pueda diseñar, construir e implementar un huerto automatizado. Esperamos pueda sacar el máximo provecho de estas iniciativas para implementar en sus casas o en sus establecimientos.

El circuito con este tipo de montaje se puede observar en la Figura 1.

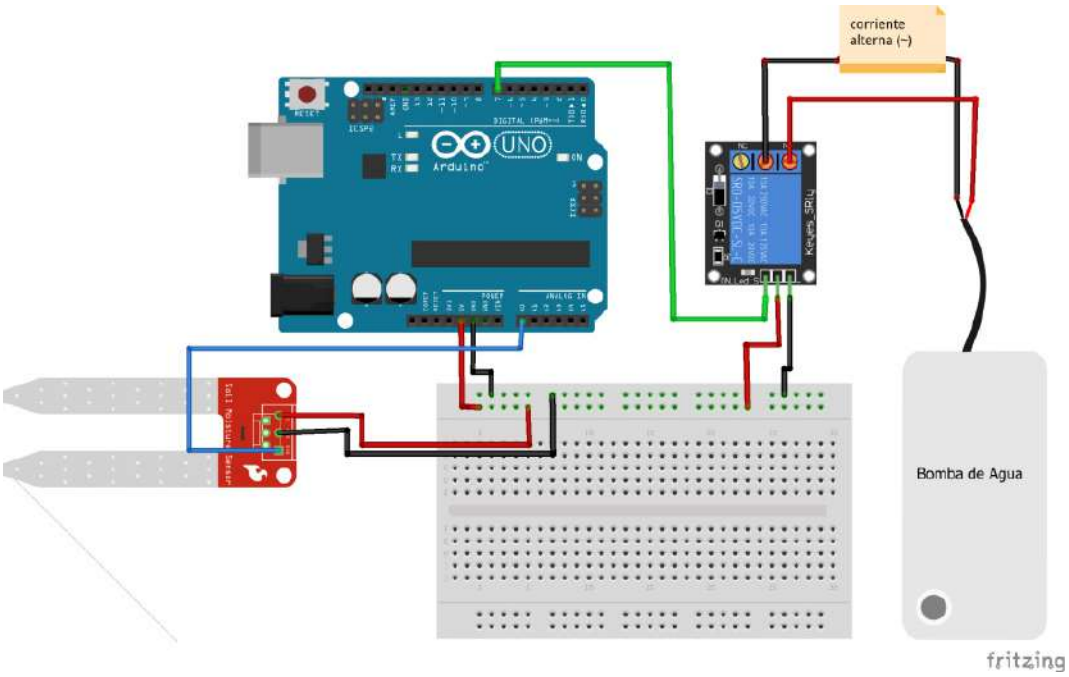


Figura 1. Esquema del circuito de riego automatizado.

Fase 1

Construcción del huerto donde se implementará el sistema de riego automatizado. Esto en fase inicial sirve para tener las dimensiones del espacio para el regadío y obtener noción de las longitudes y disposición de las mangueras a implementar.

Fase 2

La primera tarea consiste en programar el sensor de humedad de tierra con el Arduino, y explorar los valores obtenidos para ciertas condiciones de humedad, desde la tierra en la cual insertará el sensor.

Fase 3

Programar las condiciones en las cuales se activará el pin que manda el relé para que active la bomba de riego. Las bombas de agua para proyectos las puede encontrar en diversos modelos, dependiendo de la altura de agua que será capaz de levantar (presión).

Un relé en un dispositivo electromagnético que hace las veces de interruptor. En este circuito, su función básica es ser un controlador electromecánico de la intensidad de corriente que alimenta la bomba de agua (ver conexiones en Figura 2). Es necesario notar que un cable va directo a la conexión de la red pública y otro va al dispositivo y del dispositivo a la red pública (ver Figura 1).

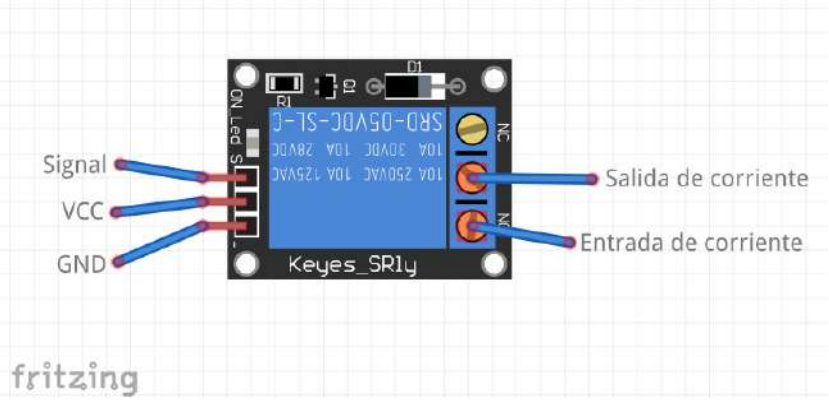


Figura 2. Conexiones del Arduino a relé, y de red pública a relé.

Fase 4 (opcional)

Programar una pantalla de lcd para observar los valores de la humedad de la tierra. Así tendrá cuantitativamente los valores de humedad mientras se realiza el riego automatizado.

Desafíos

1. ¿Qué tipo de semilla puede plantar en función de las temperaturas de esta estación?
2. ¿Cuánta humedad es la ideal para plantas, para poder calibrar el sensor y riego automatizado?

PROYECTO 7

Solmáforo **UV**

Alerta: Este proyecto involucra condiciones de exposición a peligro de electrocución, por lo cual debe ser asesorado necesariamente por un adulto con extrema precaución.

En este proyecto se propone construir un Solmáforo UV, dispositivo capaz de medir la radiación Ultravioleta (UV) y emitir una alerta por medio del código de colores de radiación UV.

Para la realización de este taller, es necesario disponer de algunos componentes electrónicos y seguir el proceso de construcción que se indica a continuación:

Componentes

- 1 protoboard
- 1 Arduino UNO
- 1 sensor GUVA S12SD (sensor UV)
- 5 relés
- 5 ampolletas (verde, amarilla, naranja, roja y violeta)
- Cables para conexiones

Actividades previas relacionadas

- **Taller 6:** Radiación UV

Desarrollo

Para el desarrollo de este proyecto se debe recordar el código de colores de radiación UV visto en el Taller 6, y los valores de voltajes que se obtendrán al conectar el sensor UV, para luego poder interpretarlos en función del código de colores (ver Figura 1).

UV Index	0	1	2	3	4	5
Vout(mV)	<50	227	318	408	503	606
Analog Value	<10	46	65	83	103	124
UV Index	6	7	8	9	10	11+
Vout(mV)	696	795	881	976	1079	1170+
Analog Value	142	162	180	200	221	240

Figura 1. Niveles de radiación y su equivalencia en los datos del sensor GUVA.

Para el diseño del circuito se propone el esquema mostrado en la Figura 2.

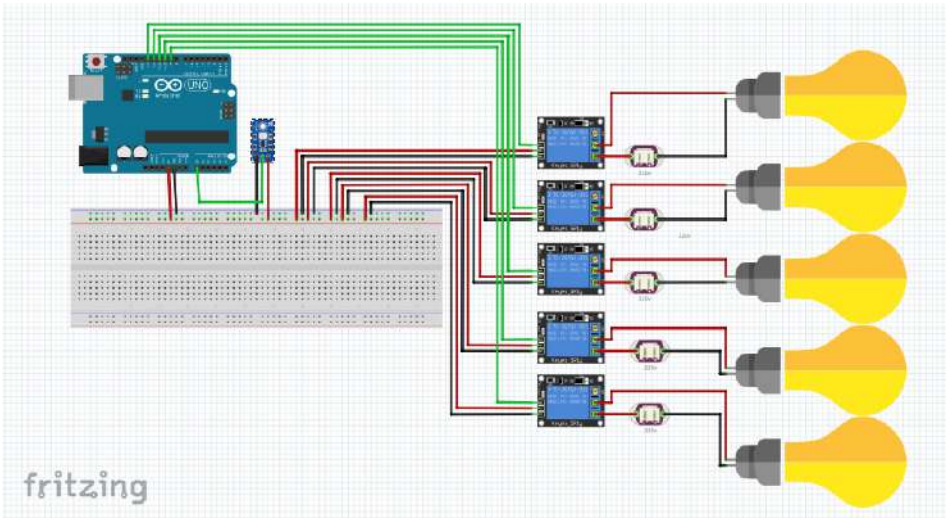


Figura 2. Esquema de circuito del solmáforo.

Fase 1

Coloque el sensor GUVA S12SD en la protoboard y conéctelo a la placa Arduino (ver Figura 3).

El circuito que corresponde al sensor GUVA S12SD se conecta a OUT, que es el pin de transmisión de datos, mediante el cual el sensor enviará datos a la placa Arduino, que los interpretará para posteriormente encender o apagar las luces. Este pin se conectará con la placa Arduino y al pin analógico 0 (pin A0). A continuación, conecte la línea de alimentación a 5V de Arduino con VCC (+) del sensor, y finalmente conecte GND de la placa con GND del sensor (-). En la figura 3 se muestran las partes principales del sensor.

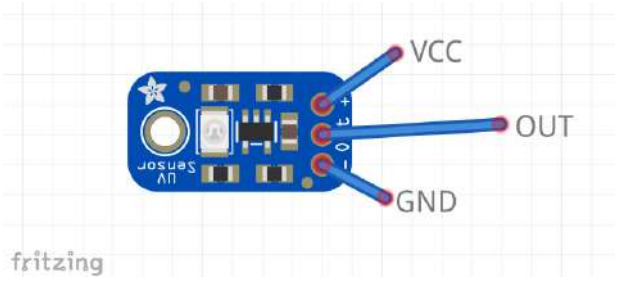


Figura 3. Componentes principales del sensor GUVA S12SD.

Fase 2

Coloque los relés en la protoboard y conéctelos a la placa Arduino. Usar como referencia la figura 2. Los circuitos que corresponden a los relés se conectan a los pines digitales 9, 10, 11, 12 y 13 del Arduino y alimentado por GND (tierra) y VCC (5V). En la figura 4 se pueden observar las conexiones principales del relé.

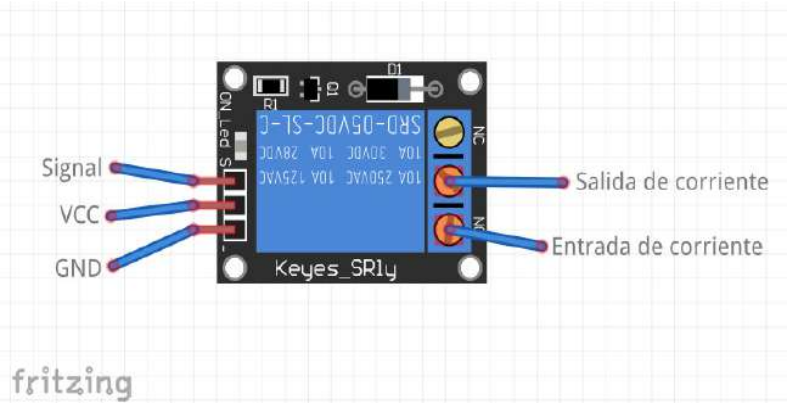


Figura 4. Conexiones principales del relé.

Fase 3

Por último, coloque las ampolletas en los relés respectivos para cada color (verde al relé 9, amarillo al 10, naranja al 11, rojo al 12 y violeta al 13) y conéctelo a la corriente como se muestra en la figura 2. A continuación, conecte la línea de alimentación a 5V de Arduino con VCC (+) del sensor, y finalmente conecte GND de la placa con GND del sensor.

A continuación, se muestra el código ejemplo que permite operar el Solmáforo UV.

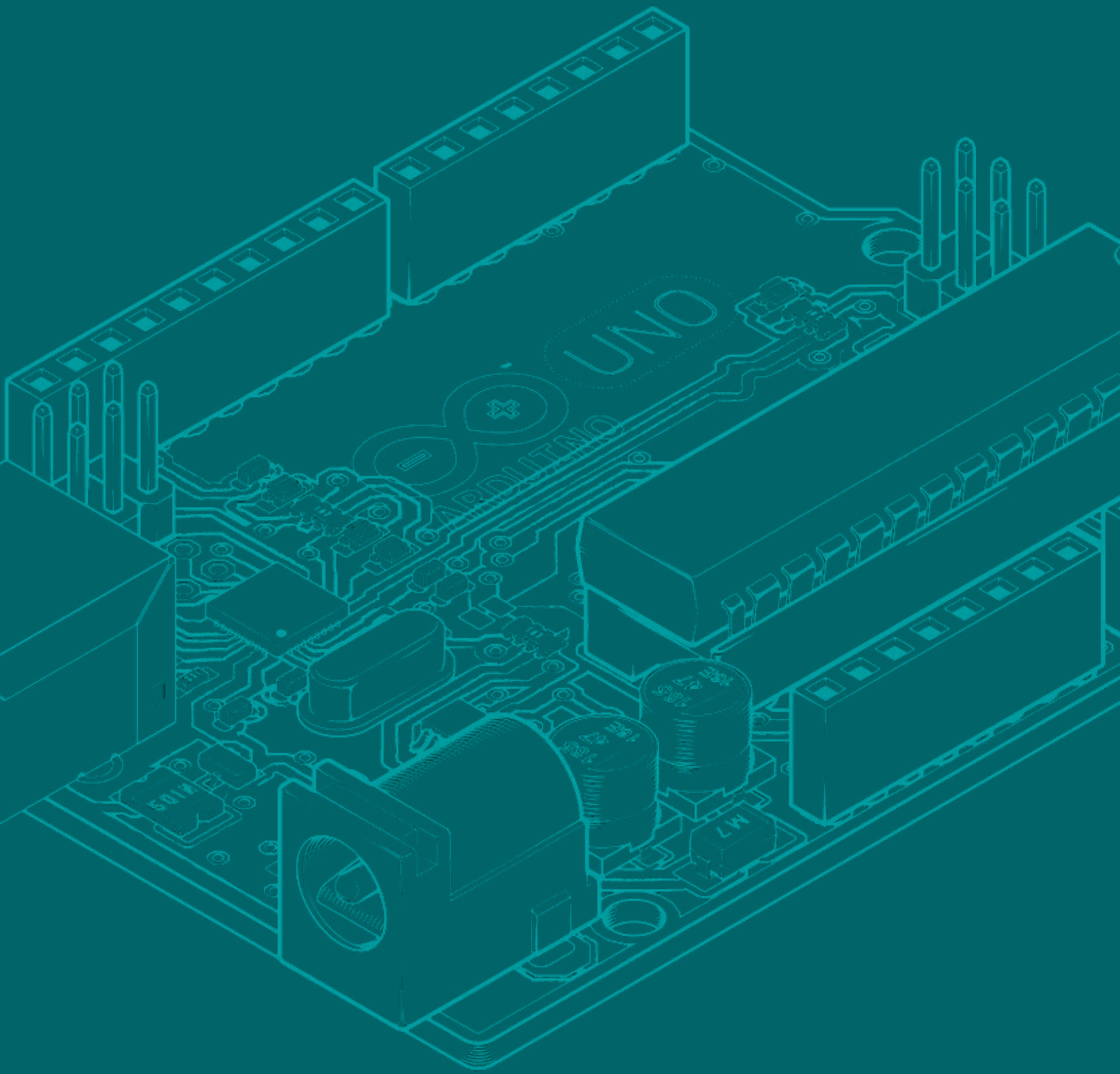
Código propuesto Solmáforo UV

```
//el programa empieza

//se definen los puertos correspondientes a cada ampolleta
int luz_verde=9;
int luz_amarilla=10;
int luz_naranja=11;
int luz_roja=12;
int luz_violeta=13;

void setup() {
pinMode(luz_verde, OUTPUT); //se inicia la ampolleta verde en la puerta de salida 9
pinMode(luz_amarilla, OUTPUT); //se inicia la ampolleta amarilla en la puerta de salida 10
pinMode(luz_naranja, OUTPUT); //se inicia la ampolleta naranja en la puerta de salida 11
pinMode(luz_roja, OUTPUT); //se inicia la ampolleta roja en la puerta de salida 12
pinMode(luz_violeta, OUTPUT); //se inicia la ampolleta violeta en la puerta de salida 13
Serial.begin(9600);
}

void loop() {
float valor; //valor entregado para calcular el voltaje
float voltaje; // voltaje del sensor
```

```
int nivel;

 valor = analogRead(A0);
 voltaje = valor/1024*5.0;

 //el voltaje se interpreta como nivel de radiación UV
 if (voltaje<408){
 Serial.print("Nivel UV bajo");
 Serial.println("");
 digitalWrite(luz_verde,HIGH); //se enciende la luz
 delay(300000); //espera 300 segundos, equivale a 5 minutos
 digitalWrite(luz_verde,LOW); //se apaga la luz
 }
 else if (voltaje>=408 && voltaje<696){
 Serial.print("Nivel UV medio");
 Serial.println("");
 digitalWrite(luz_amarilla,HIGH); //se enciende la luz
 delay(300000); //espera 300 segundos, equivale a 5 minutos
 digitalWrite(luz_amarilla,LOW); //se apaga la luz
 }
 else if (voltaje>=696 && voltaje<881){
 Serial.print("Nivel UV alto");
 Serial.println("");
 digitalWrite(luz_naranja,HIGH); //se enciende la luz
 delay(300000); //espera 300 segundos, equivale a 5 minutos
 digitalWrite(luz_naranja,LOW); //se apaga la luz
 }
 else if (voltaje>=881 && voltaje<1170){
 Serial.print("Nivel UV peligroso");
 Serial.println("");
 digitalWrite(luz_roja,HIGH); //se enciende la luz
 delay(300000); //espera 300 segundos, equivale a 5 minutos
 digitalWrite(luz_roja,LOW); //se apaga la luz
 }
 else if (voltaje>=1170){
 Serial.print("Nivel UV EXTREMO");
 Serial.println("");
 digitalWrite(luz_violeta,HIGH); //se enciende la luz
 delay(300000); //espera 300 segundos, equivale a 5 minutos
 digitalWrite(luz_violeta,LOW); //se apaga la luz
 }
 }
```

Desafíos

1. Realizar una medición del nivel de radiación UV en distintos lugares, tales como distintas salas, zonas del patio, gimnasio, etc. Luego, comparar los resultados.
2. Dados los niveles de radiación UV, ¿será posible indicar el nivel de radiación en conjunto al indicativo lumínico?



Universidad Central
Facultad de Ingeniería y Arquitectura

UMCE
el poder transformador de la educación

FinarqUcen
@Finarq_UCEN
@finarq.ucen