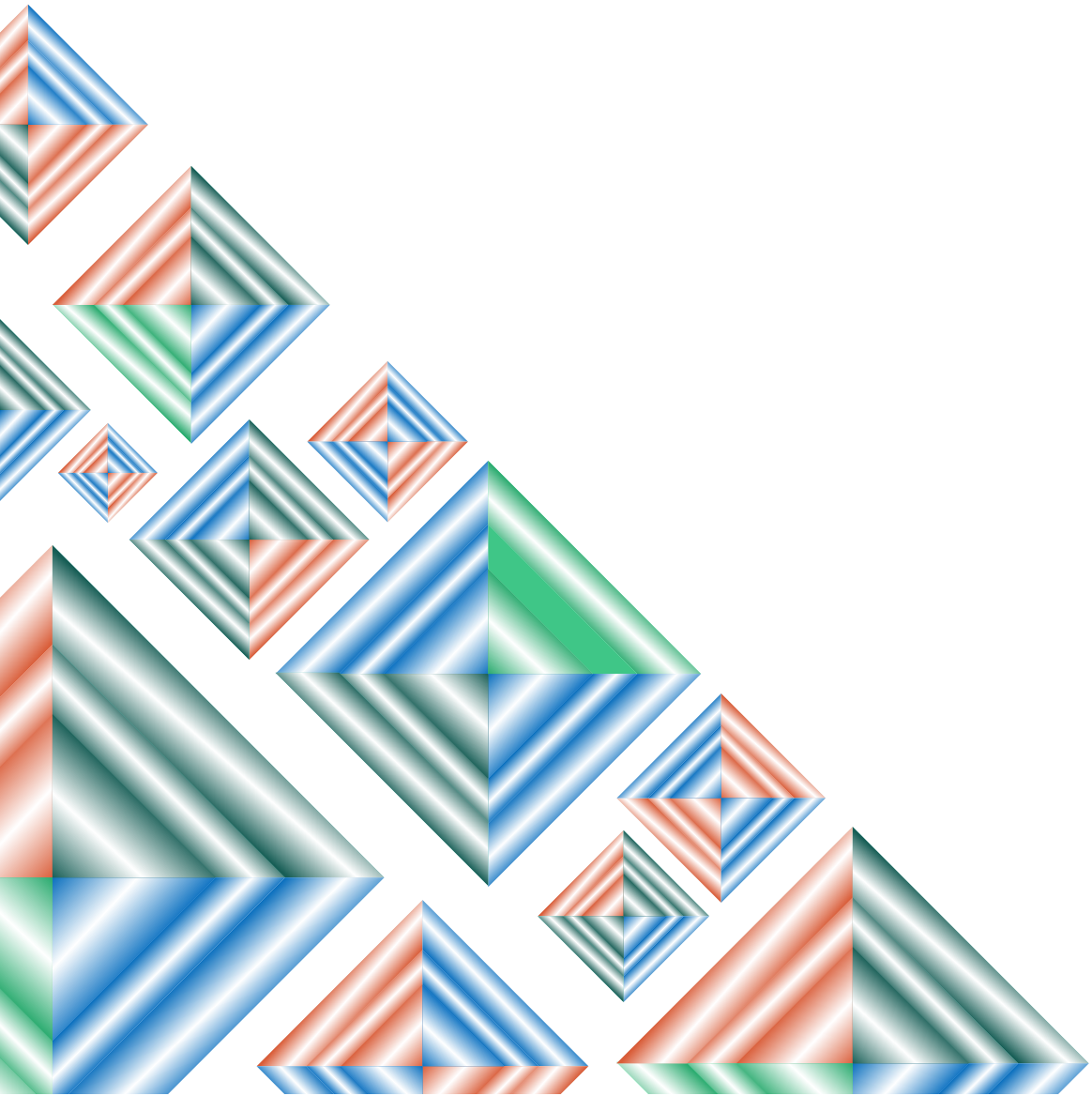
Educación STEAM

Unidad didáctica para docentes de matemáticas en formación implementando Arduino



Esta unidad didáctica fue diseñada y creada como parte del Proyecto Final de Graduación para obtener el título de Licenciatura en Enseñanza de la Matemática con Entornos Tecnológicos durante el I semestre del periodo lectivo 2024 en el Instituto Tecnológico de Costa Rica

Autores:

Bach. José Pablo Calderón Gairaud
Bach. Cristhian Alexis Chavez Montoya

Comité Evaluador:

M. Sc. Rebeca Solís Ortega (Tutora)
M. Sc. Carlos Monge Madriz (Lector)
M. Sc. Greivin Ramírez Arce (Validador)
M. Sc. Marta Vílchez Monge (Validadora)
M. Sc. Juan Carlos Brenes Torres (Validador)
Dra. Cindy Calderón-Arce (Validadora)

Agradecimientos

Se extiende un agradecimiento a todas las personas que no solo aportaron en la formación de conocimientos, sino también en el desarrollo personal de los autores. A nuestros profesores y mentores, por su guía y sabiduría; a nuestros compañeros de estudio, por su apoyo y colaboración; y a nuestras familias y amigos, por su amor y comprensión a lo largo de este proceso. Su contribución ha sido invaluable para la culminación de este trabajo.

índice

Índice de tablas

Índice de figuras

Índice de esquemas

Educación STEAM

La tendencia de la educación STEAM se plantea como una estrategia para fomentar habilidades digitales y un conocimiento sostenido en todas las áreas científicas y académicas. Su valor educativo radica en su capacidad para influir en diversas formas de abordar la realidad, con la participación dinámica de actores multidisciplinarios. Esto abre oportunidades de mejora en lo social, económico, cultural y educativo, al introducir nuevos elementos, escenarios, recursos, talentos y competencias. Además, se centra en resolver problemas del contexto real y colaborativo del aprendizaje, adaptable a distintos niveles y modalidades educativas (Aguirre et al., 2019).

En este contexto, Mendiola (2018) destaca la relevancia de la implementación de este modelo educativo dinámico, que incorpora no solo las disciplinas científicas (ciencia, tecnología, ingeniería y matemáticas), sino también las humanidades (artes y diseño), agregando un valor adicional. Este enfoque fomenta la creatividad y la interdisciplinariedad, y asume la intuición como un elemento fundamental para la realización de proyectos artísticos, científicos y descubrimientos personales cargados de imaginación.

En resumen, la educación STEAM se presenta como un enfoque educativo vital para el desarrollo integral de los individuos en la actualidad. Al combinar disciplinas científicas y humanísticas, este enfoque no solo promueve el pensamiento crítico y la resolución de problemas, sino que también fomenta la creatividad y la colaboración interdisciplinaria. Además, su adaptabilidad a diversos niveles y modalidades educativas garantiza su accesibilidad y relevancia en distintos contextos. En última instancia, la educación STEAM no solo prepara a los estudiantes para enfrentar los desafíos del futuro, sino que también los empodera para convertirse en ciudadanos activos y capaces de contribuir de manera significativa a la sociedad global.

Realidad STEAM en Costa Rica

Los datos recolectados por el noveno Estado de la Educación (Consejo Nacional de Rectores, 2023), reveló que los estudiantes de carreras STEM a menudo carecen de hábitos de estudio y enfrentan dificultades en la transición de la secundaria a la universidad, lo que implica una creciente cantidad de egresados de la educación secundaria se encuentren con habilidades insuficientes para una inserción laboral exitosa, pues no ingresan a la educación superior (agravado por la falta de acceso) ni egresan de la educación técnica del MEP. Ante este panorama, el fortalecimiento de la educación en STEM y el desarrollo del perfil laboral en estas áreas se posicionan como prioridades nacionales para impulsar el crecimiento productivo del país.

Este noveno informe también destaca los persistentes bajos rendimientos de las mujeres en pruebas de Matemática y Ciencias, lo que las coloca en una desventaja significativa en términos de oportunidades laborales y académicas en comparación con sus pares masculinos, resultando en una subrepresentación femenina en carreras y roles relacionados con STEM. A pesar de un aumento en la graduación femenina en STEM en la última década, especialmente en áreas específicas, como las ciencias sociales, la participación femenina en campos como ingeniería y computación sigue siendo limitada. Esto se refleja en el mercado laboral, donde la proporción de mujeres en ocupaciones científico-tecnológicas es inferior al promedio nacional de participación laboral femenina (Consejo Nacional de Rectores, 2023).

Además, estudios como el de Román et al. (2023) señalan que las mujeres que tienen una mayor probabilidad de seguir carreras STEM enfrentan barreras que requieren una convergencia compleja de factores, incluyendo la vocación temprana por estas materias, la confianza en sí mismas, el apoyo y estímulo del entorno familiar y educativo, y la presencia de modelos a seguir o mentores a lo largo de su trayectoria. Estudios adicionales destacan que las decisiones de las mujeres en cuanto a las carreras que eligen, y que frecuentemente están menos vinculadas con STEM, no son resultado de diferencias biológicas, sino de procesos de socialización de género que influyen

desde la infancia, evidenciando la necesidad de abordar estas dinámicas para lograr una mayor equidad en el acceso y la participación de las mujeres en campos STEM (Consejo Nacional de Rectores, 2023).

En resumen, el mercado laboral sigue privilegiando a los graduados universitarios en términos de empleabilidad y salarios, especialmente aquellos con formación en STEM, lo que perpetúa las disparidades salariales entre hombres y mujeres incluso dentro de las mismas áreas de conocimiento. Considerando que el futuro económico estará intrínsecamente ligado a las habilidades STEM, la creciente brecha entre aquellos con y sin estas habilidades podría conducir a una exclusión tanto del mercado laboral como de los sectores económicos más dinámicos. Por tanto, la promoción de la educación en STEM y la reducción de las disparidades de género en estos campos son fundamentales para garantizar una participación equitativa en la economía del futuro y evitar la marginalización de grupos específicos en la sociedad (Consejo Nacional de Rectores, 2023).

Estrategia de educación STEAM que propone el MEP

En la actualidad, el MEP tiene como objetivo principal de la Estrategia de Educación STEAM promover en los centros educativos el desarrollo de habilidades y competencias del siglo XXI en el estudiantado, desde un enfoque de género, para que exploren y valoren las áreas STEAM en sus proyectos vocacionales. Para ello plantean promover la formación y actualización del personal y técnico docentes en el desarrollo de la estrategia STEAM. Para lograr esto, el MEP categoriza una serie de habilidades que se detallan en la tabla 1.

Tabla 1

Habilidades para una educación STEAM propuestas por el MEP

Habilidad	Descripción
H1: Pensamiento Crítico	Se parte de la interrogante, ¿qué puedo crear? y se construye a partir de la experiencia, no de la transferencia de conocimiento, lo cual requiere comprensión, análisis, interpretación, evaluación, lo que promueve la autorregulación del aprendizaje
H2: Creatividad	Brinda la posibilidad de materializar las ideas y construir conocimiento desde un proceso activo donde interviene la imaginación y la curiosidad
H3: Colaboración	Fomenta el trabajo en equipo y aprender a tomar decisiones de manera conjunta
H4: Investigación	Plantean hipótesis y realizan experimentación para la solución de problemas, favoreciendo el aprendizaje a largo plazo de conceptos
H5: Comunicación	Al ser un trabajo en equipo se impulsan las capacidades expositivas de comunicación, transmisión de ideas, negociación entre otras
H6: Solución de Problemas	Desarrolla la capacidad de resolución de problemas de manera creativa, así como el desarrollo de la gestión emocional y pensamiento lógico matemático

Nota: Adaptado de Manual Interactivo para la ruta de trabajo "Educación STEAM", MEP, 2023.

Se puede notar que estas habilidades son relevantes en todas las áreas, aunque en algunas se enfatice más una que otra. Sin embargo, son fundamentales en todo proceso de aprendizaje constructivista, que se centra en resolver problemas e investigar, siendo esta la base del enfoque STEAM.

Sobre la unidad didáctica

La presente unidad didáctica, centrada en Arduino, representa una estrategia innovadora para cultivar el desarrollo de habilidades STEAM (ciencia, tecnología, ingeniería, arte y matemáticas) para con sus usuarios, es decir, profesores de matemáticas en formación. Ante el auge tecnológico, es crucial equipar a los educadores con las herramientas y el conocimiento necesarios para integrar

eficazmente conceptos STEAM en sus prácticas pedagógicas. Arduino, como plataforma de código abierto, proporciona un entorno accesible y versátil para la experimentación y el aprendizaje práctico en áreas clave de la educación STEAM.

Al implementar una unidad didáctica en torno a Arduino, los usuarios tienen la oportunidad de involucrarse en proyectos interdisciplinarios que integran conceptos de programación, electrónica, diseño y resolución de problemas. Este enfoque holístico les permite no solo adquirir conocimientos técnicos, sino también desarrollar habilidades como la creatividad, el pensamiento crítico y la colaboración, que son fundamentales en el mundo actual. Además, se busca transformar el aula, promoviendo un enfoque activo y participativo del aprendizaje, donde los estudiantes no solo adquieren conocimientos teóricos, sino que también tienen la oportunidad de aplicarlos en situaciones prácticas y así trascender de la sola trasmisión de conocimiento (Segarra-Vera et al., 2024).

La unidad didáctica se encuentra estructurada en tres secciones principales: Introducción a Arduino, una guía sobre el funcionamiento y componentes por utilizar a lo largo de la unidad; sesiones de trabajo, actividades prácticas programables con Arduino donde se pretenden desarrollar a pertinencia las habilidades para una educación STEAM mostradas en la tabla 1; y por último el epílogo, que trata sobre consideraciones finales que los autores idealizan en una posterior ejecución de la unidad y reflexiones sobre la creación de este material.

i. Sobre las sesiones de trabajo

En total son 8 sesiones de trabajo, cada una de ellas está compuesta por actividades o tareas referentes al componente que se va a utilizar en dicha sesión. La última representa una culminación de aprendizaje en la que se debe aplicar el conocimiento adquirido en las sesiones anteriores para, creativamente, usar Arduino como apoyo en el proceso de aprendizaje de un contenido matemático y enlazar los conocimientos didácticos con las funcionalidades del microcontrolador. De esa forma, se demuestra una aplicabilidad educativa y el interés de esta unidad didáctica.

Las sesiones de trabajo pretenden desarrollar habilidades STEAM en los docentes de matemáticas en formación por medio de tareas con diferentes niveles de dificultad en cada sesión. Una vez que los usuarios han adquirido los conocimientos esenciales, se visiona una integración interdisciplinaria con su formación docente, que abarca áreas de conocimiento matemático, didáctico y pedagógico. Los educadores podrán fusionar sus conocimientos didácticos y pedagógicos con la aplicación práctica de conceptos STEAM, lo que enriquecerá la experiencia educativa y promoverá un aprendizaje más profundo y significativo para los estudiantes.

En cada sesión se presenta una pequeña descripción que ubica al lector en su aprendizaje y se detallan tres aspectos a considerar antes de empezar cada sesión de trabajo: el tiempo aproximado de duración sugerido, el listado de tareas específicas y las habilidades pertinentes de estudio. Un ejemplo de la distribución se presenta en la tabla 2.

Tabla 2

Ejemplo de aspectos técnicos por sesión

0:00 horas	H1-H2-H3-H4-H5-H6
Tarea 1	
Tarea 2	
Tarea 3	

ii. Sobre las tareas específicas

Cada tarea relacionada al uso de Arduino tiene una estructura base donde se describe el objetivo de esta, materiales por utilizar, se presenta un esquema con las conexiones que deben realizarse, una guía paso a paso remarcada en un cuadro gris para llevar a cabo la actividad y el código que debe implementarse para dicha tarea.

Cada sesión de trabajo presenta una tarea denominada “Trabajo Independiente” que tiene como objetivo estimular la investigación y funcionar como una herramienta de autoevaluación para los usuarios, permitiéndoles aplicar y profundizar en los conocimientos adquiridos a lo largo de las sesiones. Esta tarea proporcionará a los

usuarios la oportunidad de explorar temas de interés específicos en mayor profundidad, realizar experimentos y proyectos personales, y reflexionar sobre su comprensión y habilidades. Además de fortalecer el aprendizaje individual, esta iniciativa fomentará un enfoque activo y autónomo hacia el conocimiento, impulsando así un mayor compromiso y desarrollo personal. Esta tarea no se considera dentro del tiempo sugerido para abarcar cada sesión. En la tabla 3 se resumen las 8 sesiones de trabajo y las tareas específicas para cada sesión.

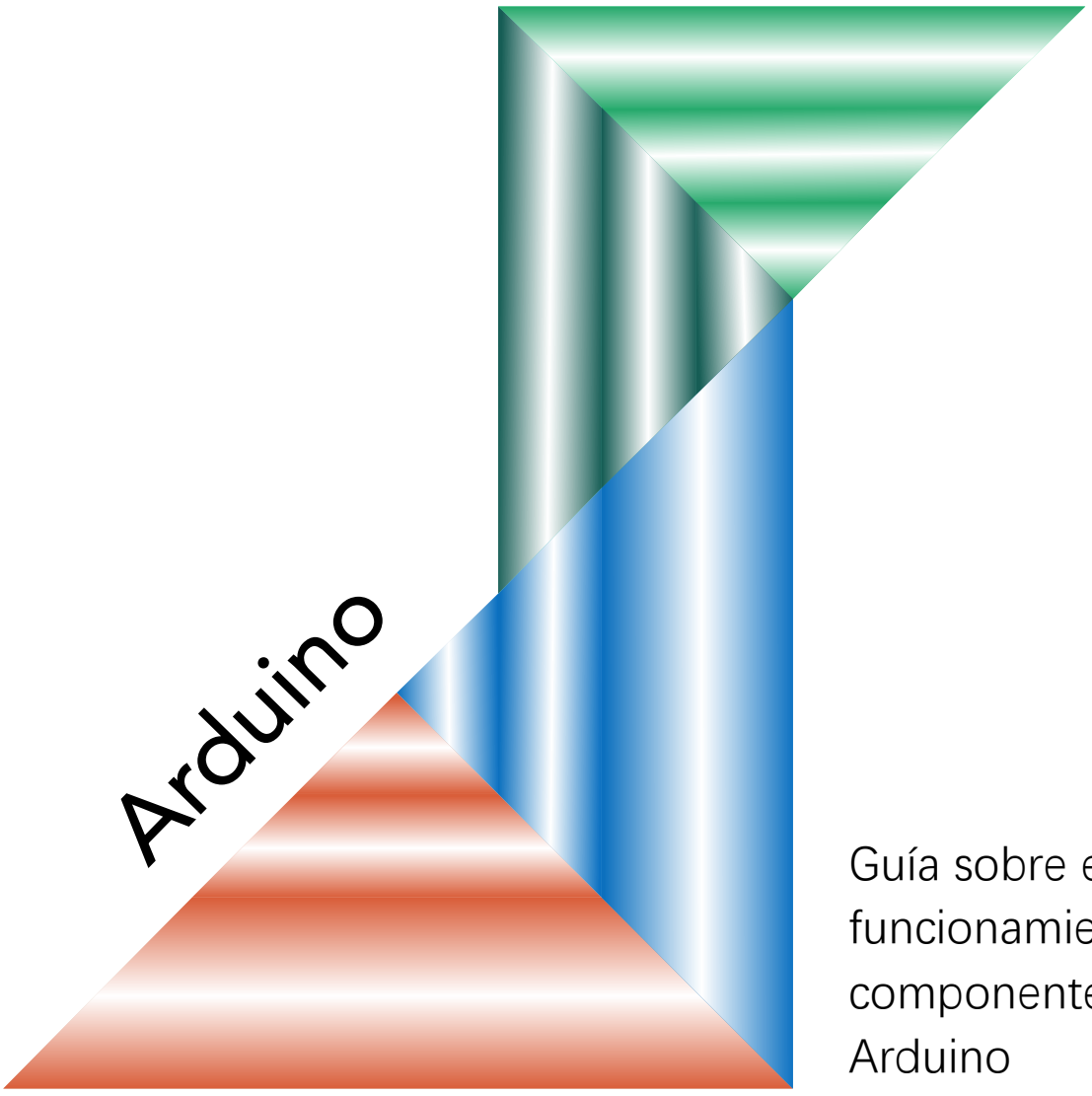
Tabla 3

Resumen sesiones de trabajo

Sesión (tiempo sugerido)	Tareas
Sesión 1 Introducción a Arduino y su funcionamiento (1:00 hora)	1. Encender una luz LED 2. Encender y apagar una luz LED 3. Secuencias de luces LED
Sesión 2 Luces LED y botones (1:30 horas)	1. Reconocer el estado del botón 2. Luz LED con botón 3. Luz LED con dos botones 4. Luces LED alternadas con botón
Sesión 3 Matriz 8x8 (1:30 horas)	1. Encender cada luz de la matriz a mano 2. Recorrer la matriz 3. Imprimir un gato 4. Escribir una frase
Sesión 4 Funcionamiento del display LCD (1:00 hora)	1. Hola mundo y contador de segundos 2. Texto deslizante 3. Display, botón y LED
Sesión 5 Sensores (1:00 hora)	1. Leer sensor de temperatura 2. Imprimir datos de sensor de temperatura en el LCD 3. Leer fotorresistor 4. Encender LED con fotorresistor 5. Leer control de luz infrarroja IDE 6. Encender LED con control de luz infrarroja

Sesión 6 Display 7-Segmentos Display 4x7 segmentos (1:30 horas)	1. Contador ascendente con display 7-segmentos 2. Número random con display 7-segmentos 3. Secuencias de números display 4x7 segmentos 4. Contador hasta 9999 modulando velocidad
Sesión 7 Motor y servomotor (1:30 horas)	1. Motor paso a paso con controlador 2. Motor paso a paso con controlador y batería 9V 3. Servomotor 4. Controlar servomotor con potenciómetro
Sesión 8 Arduino y matemáticas (1:30 horas)	1. Dado digital con matriz 8x8 2. Calculadora de sumas y restas 3. Ruleta aleatoria

La forma en que se evalúan los aprendizajes en cada sesión queda abierta a quienes implementen la unidad. Esto variará según el tiempo dedicado a la implementación (basándose en el tiempo sugerido por los autores), el número de lecciones y otros aspectos relevantes para su aplicación. Este enfoque flexible permite adaptar aspectos evaluativos a las necesidades y contextos específicos de los estudiantes, lo que facilita una evaluación más integral y auténtica del aprendizaje. Los autores sugieren la rúbrica presentada en el anexo 1 para evaluar el desenlace de cada una de las sesiones, puede sufrir cambios a disposición de quiénes y cómo deseen implementarla.



Guía sobre el funcionamiento y componentes de Arduino

Esta sección es una introducción a Arduino donde se explica qué es, cuál es su funcionamiento y cuáles componentes se utilizarán particularmente para llevar a cabo las tareas de la presente unidad didáctica. Se describe además el entorno de programación de Arduino para escribir, estructurar, cargar, entre otros aspectos, el código de programación de la placa. Por último, se muestran ejemplos de proyectos con Arduino para visualizar el potencial de esta placa.

El primer concepto que se debe tratar es el de un microcontrolador, este es un circuito integrado digital programable de memoria de solo lectura (Read Only Memory, ROM) y funcionalidad de entrada/salida (Input/Output,) que pueden programarse para varias funciones de control. Entre los microcontroladores más usados se tiene la familia de Arduino. Los circuitos integrados son circuitos eléctricos pequeños de materiales semiconductores (como el silicio) tratados adecuadamente para reunir los transistores, resistencias, condensadores, entre otros elementos y realizar una función electrónica específica, por esos se habla de proyectos eléctricos al trabajar con Arduino.

Componentes electrónicos

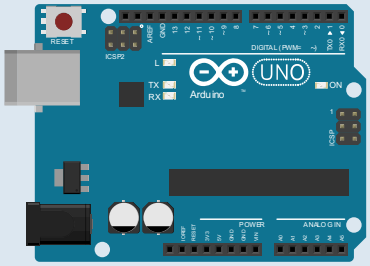
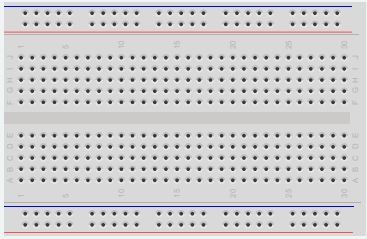
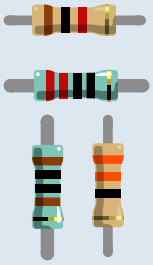
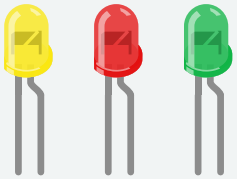
Para esta unidad didáctica se requiere de un "Starter Kit" de Arduino como el que se observa en la figura 1. El Starter Kit es un paquete físico con varios componentes básicos para llevar a cabo actividades iniciales que permiten comprender el funcionamiento de la placa Arduino y su potencial. No todos los paquetes contienen los mismos componentes, pero se pueden comprar por aparte de ser necesario. En la tabla 4 se listan y detallan los componentes que se usan a lo largo de la unidad didáctica. De igual forma, se extiende una invitación a los usuarios a investigar sobre más componentes y explotar la funcionalidad de Arduino.

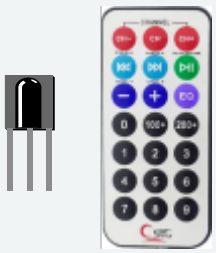
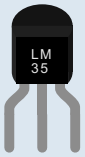
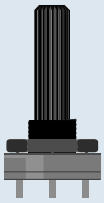
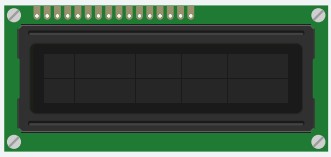
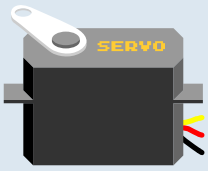
Figura 1

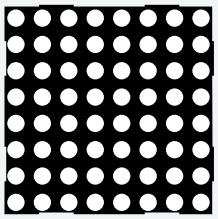
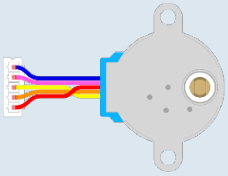
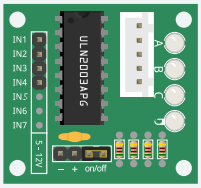
Imagen de referencia del Starter Kit

Tabla 4

Lista descriptiva de los componentes por utilizar en la unidad

Componente	Descripción
	Arduino UNO es una tarjeta electrónica que combina un microcontrolador y un conjunto de pines para conexión de entrada y salida, que permiten, mediante un programa editado con el entorno de desarrollo propio de Arduino (IDE), interaccionar con el medio físico mediante sensores y actuadores. Las partes de la placa de Arduino UNO que más se destacan en uso son el botón de reinicio, 14 pines digitales, controlador, 6 pines análogos, pines de poder, puerto de energía, puerto USB.
	La protoboard o tablero de circuitos es un tablero para realizar conexiones electrónicas, cubierta con orificios para cables y con conductores (que sirven como cables adicionales). Se puede conectar al Arduino por medio de los jumpers wires o cables puente. Con la protoboard se administran las conexiones.
	Las resistencias son un componente para resistir el flujo de energía eléctrica. Convierte parte de la energía en calor. Una resistencia termina suministrando la cantidad de energía que necesitan los demás componentes. Por ejemplo, conectar un LED sin resistencia haría que sea más brillante por unos momentos, pero luego se quemaría. La resistencia almacena la energía que sería exceso para el LED y solo deja fluir la cantidad necesaria.
	Un LED es un componente que convierte la energía eléctrica en energía luminosa. La “pata” más larga del LED se llama ánodo y se conectará a la corriente (electricidad). La “pata” más corta es un cátodo y se conectará a tierra. Cuando se aplica voltaje al ánodo del LED y el cátodo se conecta a tierra, el LED emite luz.
	Un botón o pulsador permite interrumpir o enviar una señal electrónica. Los pulsadores tienen un efecto rebote cuando se pulsan. Es decir, cuando se presiona o suelta se produce una fluctuación en la señal que puede hacer que se pase de un estado HIGH a LOW o viceversa.

	Un receptor de infrarrojos (receptor IR) utiliza comunicación por infrarrojos. La luz infrarroja, a diferencia de la luz visible, tiene una longitud de onda ligeramente más larga y, por tanto, es indetectable para el ojo humano. El control por cada botón envía una frecuencia IR diferente que es interpretada por el componente.
	Un sensor de temperatura tiene el funcionamiento de entregar información correspondiente a la temperatura del ambiente por medio del voltaje en su pin de señal. Esta unidad didáctica usa el sensor de temperatura LM35, la placa de Arduino recibe la señal y la lee con uno de sus pines analógicos de entrada.
	La forma más simple de alimentar el Arduino es a través del conector USB, este puerto brinda una alimentación de +5V. Sin embargo, una vez que se desea colocar la tarjeta Arduino en su aplicación final, el puerto USB puede no ser la forma óptima de alimentarla. Para esta unidad didáctica se usará una batería de 9V.
	Un potenciómetro es una resistencia variable. Se puede controlar la intensidad de corriente que fluye por un circuito si se conecta en paralelo y la diferencia de potencial al conectarlo en serie. Suele ser útil en proyectos que requieren cambios de intensidad eléctrica en el circuito y son accionados por el usuario. Para la unidad didáctica se utilizará el B10K.
	Un LCD Arduino es un display de cristal líquido que se utiliza para mostrar información en proyectos de Arduino. Los LCD Arduino son populares debido a su facilidad de uso y bajo costo. Hay de distintos modelos y dimensiones, para la unidad didáctica se utilizará un LCD 16x2.
	Un servomotor es un dispositivo alimentado por corriente continua que puede controlar de modo muy exacto la posición (de 0º a 180º) o la velocidad (en revoluciones por minuto, rpm, en sentido horario o antihorario). Tienen 3 pines para su conexión: alimentación (5V, normalmente), tierra (GND) y el pin de la señal.

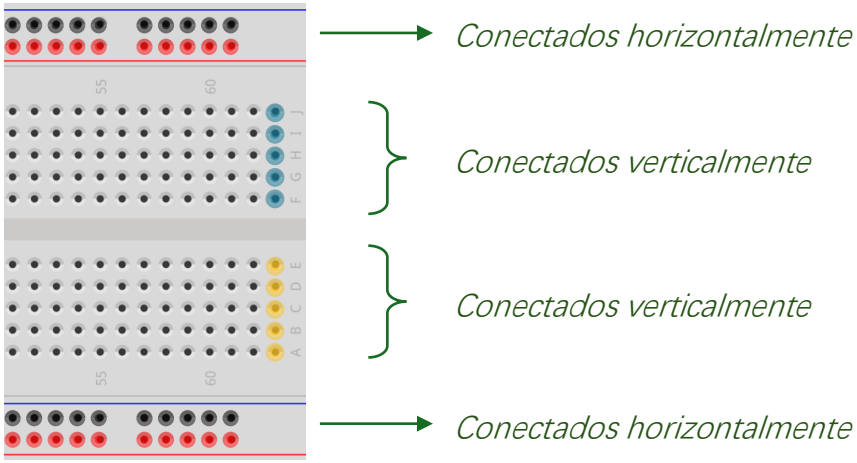
	La matriz de LED 8×8 tiene 8 filas y 8 columnas de LED controlables individualmente. Las matrices de LED pueden utilizarse para carteles publicitarios, visualización de la temperatura y otras aplicaciones.
	Existen varios tipos de motores compatibles con Arduino. Para la unidad didáctica se necesitará un motor 28BYJ-48, el cual es un motor paso a paso unipolar, lo que significa que la vuelta completa se puede dividir en “pasos” lo que permite controlar la dirección y ángulo de giro.
	Un controlador (driver) de motor es un pequeño amplificador de corriente; la función de los drivers de motor es tomar una señal de control de baja corriente y luego convertirla en una señal de corriente más alta que pueda conducir un motor.
	Un fotorresistor CdS es un componente electrónico cuya resistencia cambia según el brillo. El sulfuro de cadmio (CdS) se utiliza como material interno y su valor de resistencia disminuye a medida que la luz que lo incide se vuelve más fuerte.
	Los cables puente o jumpers wires son simplemente cables que tienen pines conectores en cada extremo, lo que permite usarlos para conectar dos puntos entre sí sin soldar. Los jumpers wires se utilizan normalmente con la protoboard para facilitar el cambio de un circuito según sea necesario.
	Un display 7-segmentos es un componente que tiene 7 segmentos LED, más un LED que hará de punto. Por lo tanto, se debe trabajar como si se tuvieran 7 LED conectados al Arduino.
	El display 4x7-segmentos se suele utilizar para crear contadores o temporizadores en el Arduino y para la salida de valores digitales de los sensores en los proyectos. Funciona similar a un Display 7-segmentos, pero aplicado a los 4 dígitos.

Es necesario mencionar que, en proyectos eléctricos, hay dos términos importantes: power (energía) y ground (tierra). En circuitos, el cable de alimentación (energía) es de donde proviene la electricidad y el cable de tierra es su camino de

retorno. En la protoboard, la energía y tierra tienen sus propios “canales” para conectarlos al Arduino. Sin embargo, no es obligatorio conectarlos en esos canales, lo más importante es respetar cómo están conectados los canales en la protoboard como se observa en la figura 2.

Figura 2

Esquema de conexiones eléctricas en la protoboard



En la placa Arduino existen dos tipos de pines, los pines de entrada (conocidos como input pin) y los de salida (conocidos como output pin). Un pin de entrada en una placa Arduino es un pin digital o analógico que se utiliza para recibir información de dispositivos o sensores externos. Puede detectar el estado de un botón, un sensor o cualquier otro dispositivo que envíe una señal a la placa. Se puede configurar un pin de entrada para leer una señal digital o analógica.

Por otro lado, un pin de salida en una placa Arduino es un pin digital que se utiliza para enviar información a dispositivos externos como un LED, un motor o cualquier otro dispositivo que requiera una señal para funcionar. Cuando un pin de salida se establece en ALTO, envía un voltaje de 5V, mientras que al configurarlo en BAJO envía 0V.

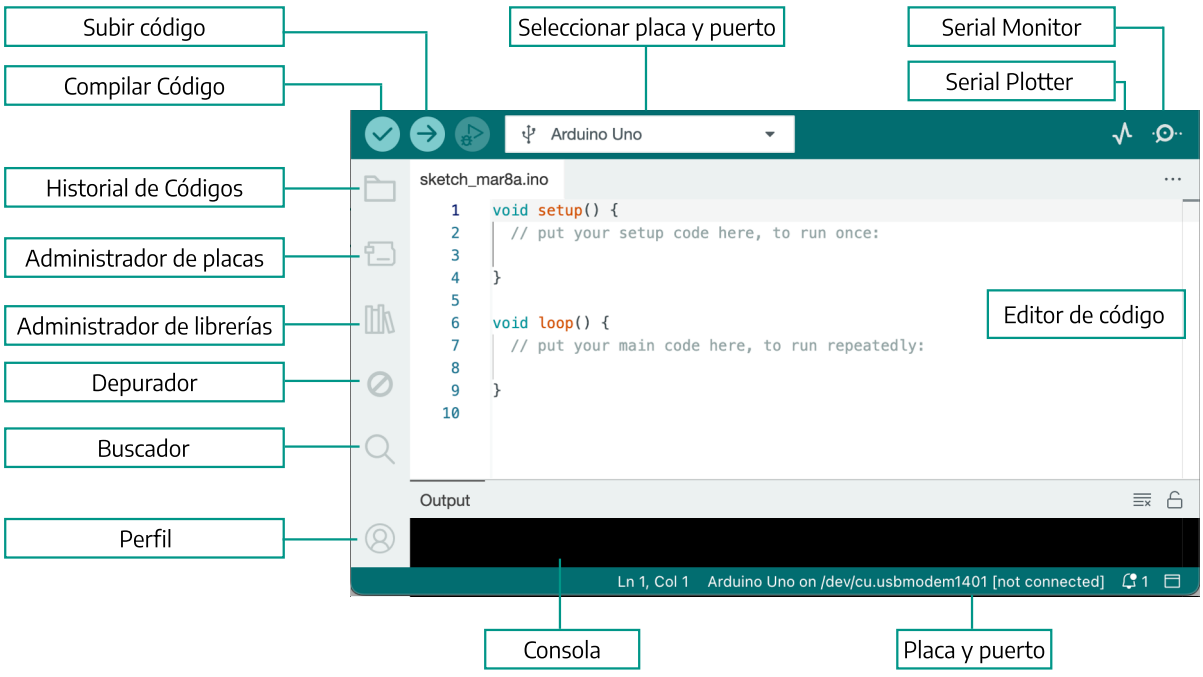
Entorno de programación Arduino IDE

Antes de aprender sobre cómo programar la placa Arduino, se necesita conocer su entorno de programación; para ello, se necesita tener instalado en el ordenador el entorno de desarrollo integrado (IDE) de Arduino que puede obtenerse de la página oficial colocando en el buscador de tú navegador preferido: “Instalar Arduino IDE”.

El IDE es una aplicación multiplataforma que está escrita en el lenguaje de programación Java y se utiliza para escribir y cargar programas en placas compatibles con Arduino. Para descargarla, basta con ingresar a la página oficial de Arduino y seleccionar la versión más reciente para su sistema operativo. Una vez instalada y abierta la aplicación, se abrirá una pestaña como la que se presenta en la figura 3.

Figura 3

Partes del entorno de programación de Arduino



A continuación, se describe la función de cada apartado:

- Serial Monitor: permite enviar y recibir mensajes de texto de la placa Arduino
- Serial Plotter: grafica las variables obtenidas de la placa Arduino
- Seleccionar placa y puerto: permite seleccionar la placa Arduino y el puerto desde el cual se va a utilizar

- Subir código: una vez escrito el código, permite enviarlo a la placa Arduino
- Compilar código: antes de subir el código se debe compilar en el ordenador para verificar que no haya errores de programación
- Historial de códigos: permite visualizar los borradores de códigos anteriores
- Administrador de placas: algunas placas de Arduino requieren de instalaciones extras que se pueden obtener en esta opción
- Administrador de librerías: algunos componentes requieren de librerías para usarlos con mayor facilidad, las librerías se pueden obtener en esta opción
- Depurador: permite dar seguimiento a las variables y al código paso a paso
- Buscador: permite buscar texto específico dentro del código
- Perfil: en esta opción se puede iniciar sesión con la cuenta de Arduino
- Consola: imprime las variables y visualizar el Serial Monitor y el Serial Plotter
- Placa y puerto: confirma que la placa y puerto seleccionado se encuentran conectados
- Editor de código: en esta sección se escribe el código de programación para la placa Arduino

Al descargar el programa, Arduino permite al código participar en la inicialización del sistema. Para ello, se debe especificar al microcontrolador los comandos que ejecutará en el momento del arranque y luego olvidarse de ellos (es decir, estos comandos sólo se ejecutarán una vez al inicio del sistema).

Por lo anterior, en el programa se tiene que seleccionar el bloque en el que se almacenarán estos comandos. void setup (), o más bien el espacio dentro de los corchetes de esta función, es el lugar dentro del sketch de Arduino donde esto sucede. La función de bucle o void loop (), es la función principal, el punto de entrada al programa. Es el lugar donde se tiene que poner los comandos que se ejecutarán mientras la placa Arduino esté habilitada. Comenzando con el primer comando, el microcontrolador irá hasta el final y saltará inmediatamente al principio para repetir la misma secuencia. Y así un número infinito de veces (siempre y cuando la placa tenga

suministro eléctrico). Antes del void setup () y el void loop () se suelen definir todas las variables globales que se utilizarán en el código.

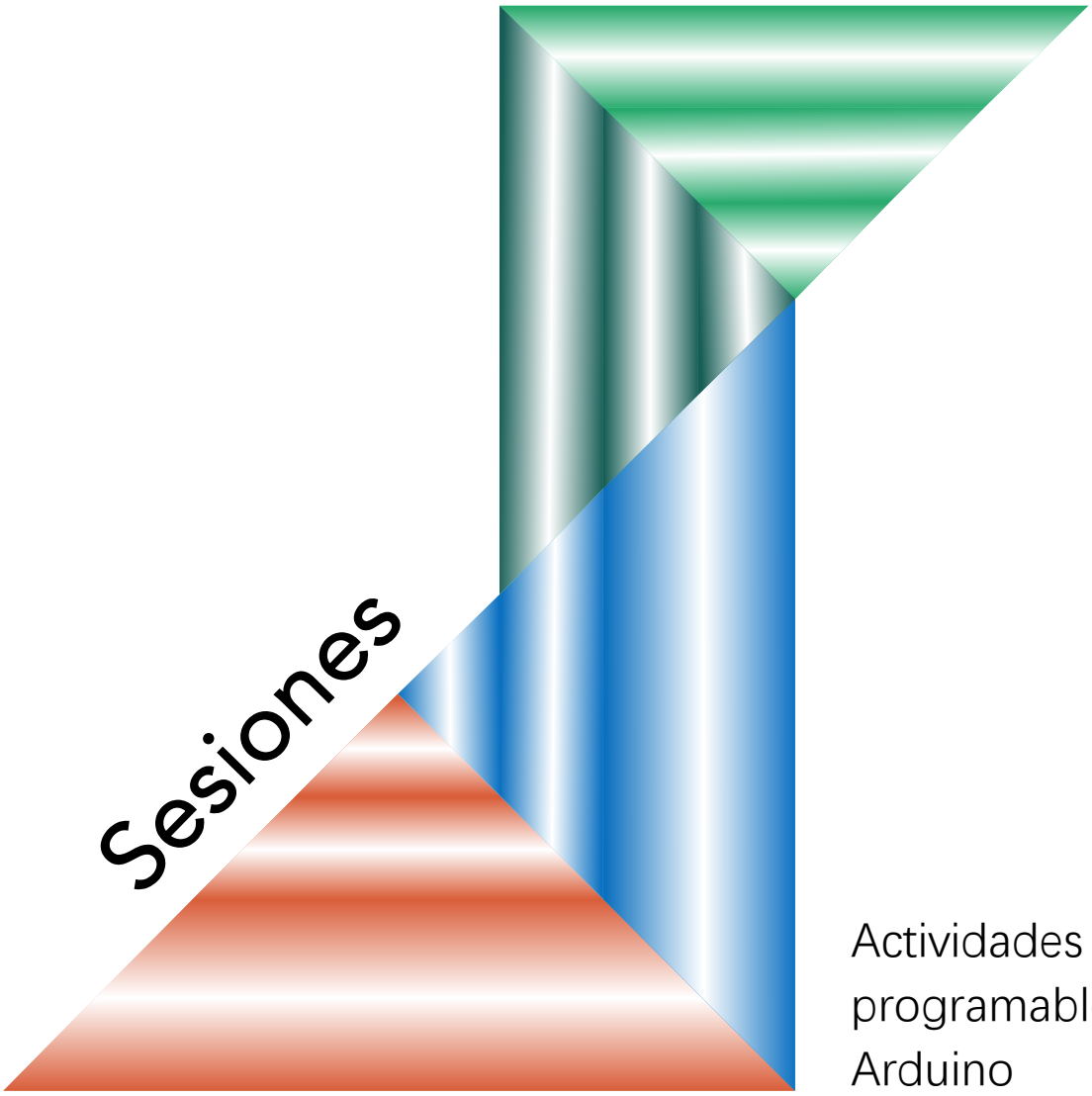
Ejemplos de proyectos con Arduino

Con Arduino, los proyectos pueden ser tan “simples” como un semáforo controlado por un microcontrolador o tan complejos como un robot autónomo. Aunque se usen componentes básicos como resistencias y sensores, las posibilidades son ilimitadas. Esto demuestra que, con creatividad y conocimiento, incluso los componentes más simples pueden dar vida a proyectos sorprendentemente complejos. ¡Escanee los siguientes códigos QR para que pueda apreciar el potencial de Arduino!

Sistema de Riego	Grúa con Servomotores	Mano Robótica	Más ideas increíbles
Elaborado por Fer Informática (2022)	*Elaborado por Robot UNO (2020)*	*Elaborado por Álvaro González (2016)*	*Elaborado por Ale Herrera (s.f.)*

Antes de llevar a cabo proyectos como los anteriores, es necesario empezar con lo básico. Para ello, se iniciará con las sesiones de trabajo. La primera sesión servirá como base para comprender el funcionamiento de Arduino, cada actividad presenta un diagrama/esquema para realizar las conexiones necesarias y llevar a cabo las tareas, se especificará el objetivo, materiales necesarios, guía paso a paso de las conexiones, y código por utilizar. Los esquemas de las conexiones electrónicas fueron realizados por los autores con el apoyo de la herramienta Fritzing, un software cuyo objetivo es fomentar un ecosistema creativo donde los usuarios puedan diseñar sus prototipos y que una de sus causalidades sea enseñar electrónica en entornos educativos.

Se recomienda al usuario ir asegurando la comprensión en las actividades y verificar que todo funciona correctamente, la guía paso a paso disminuye en complejidad de detalle pues se asume una mejor compresión a medida que avanzan las sesiones de trabajo. Además, en caso de no tener claridad en algún aspecto de las sesiones de trabajo como el funcionamiento de la placa o los componentes, se pueden aprovechar los recursos disponibles en la página oficial o la web, en particular porque Arduino tiene una gran popularidad en la comunidad en línea y la cantidad de documentación sobre su uso es basta.



Actividades prácticas
programables con
Arduino

Sesión 1

Introducción a Arduino y su funcionamiento

1:00 hora | H1-H2-H3-H4-H6

ESPECIFICACIONES

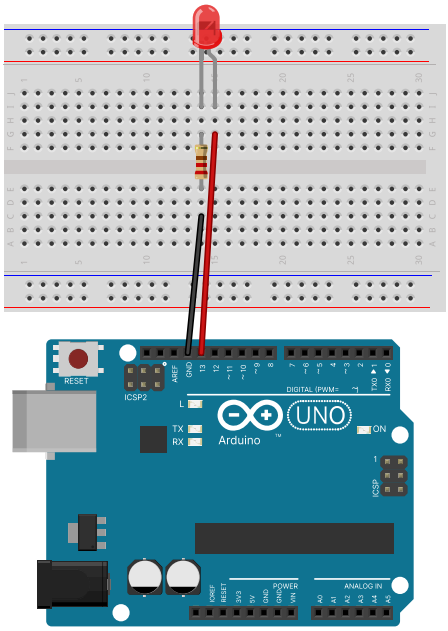
- Encender una luz LED [H1]
- Encender y apagar una luz LED [H1 – H3 – H4]
- Secuencias de luces LED [H1]
- Trabajo independiente [H1 – H2 – H4 -H6]

Arduino tiene una gran capacidad para llevar a cabo proyectos que podrían influir o aportar en la rutina/quehacer de una persona. Sin embargo, antes se debe comprender el funcionamiento o la mecánica detrás de los proyectos de Arduino. A continuación, se plantearán una serie de actividades para manipular luces LED.

i. Encender una luz LED

Esta primera actividad consiste en encender un LED mediante las salidas de Arduino, esto debe ocurrir al momento de cargar el código en la placa. El LED se mantendrá encendido mientras reciba corriente. Para realizarlo, se deben realizar las conexiones como se observa en el esquema 1.

Esquema 1: encender una luz LED



Objetivo:

Encender una luz LED.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers wires
4. Luz LED
5. Resistencia 220 Ω

Conectar un LED sin resistencia puede quemarlo.

1. Conecte el LED a la protoboard, el ánodo y cátodo deben ir en canales diferentes de la protoboard.
2. Conecte un jumper wire de uno de los pines digitales del Arduino (para nuestro ejemplo se conecta en el pin digital 13) y realiza la conexión positiva con el LED.
3. Coloque una resistencia en el mismo canal donde se encuentra el cátodo del LED.
4. Conecte un jumper wire a la terminal de tierra del Arduino al mismo canal donde se encuentra la resistencia.
5. Una vez completado el esquema, conecte el Arduino al computador y coloque el código en el IDE. ¡Cargue el código y listo! El LED debe estar encendido en la protoboard.

Código

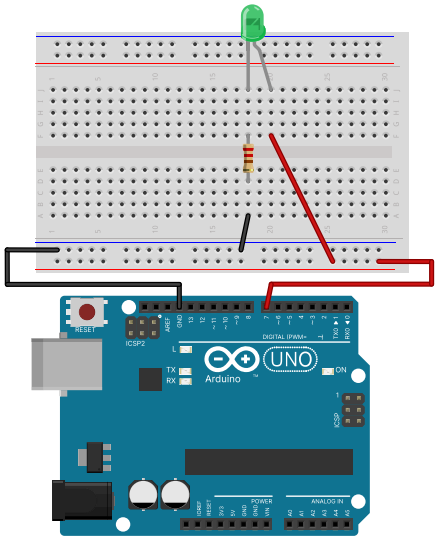
```
1. const int led=13; // Define LED como su respectivo pin
2.
3. void setup(){
4.   pinMode(LED,OUTPUT); // Define al pin como OUTPUT
5. }
6.
7. void loop(){
8.   digitalWrite(LED,HIGH); // Enciende el LED8.
9. }
```

Como se mencionó anteriormente, al cargar el código, el LED debió encenderse y mantenerse de esa forma. Esto sucederá siempre y cuando la placa de Arduino tenga corriente que en este caso está siendo obtenida del computador. En la siguiente actividad, se apagará y encenderá el LED cada un segundo.

ii. Encender y apagar una luz LED

Ahora, se puede modificar un poco el código para cambiar el estado del LED. Se podría usar el mismo esquema, sin embargo, se propone otro a continuación para manipular de forma diferente las conexiones electrónicas. Para realizarlo, se deben realizar las conexiones como se observa en el esquema 2.

Esquema 2: encender y apagar una luz LED



Objetivo:

Encender y apagar una luz LED cada segundo.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers Wires
4. Luz LED
5. Resistencia 220 Ω

1. Conecte el LED a la protoboard, el ánodo y cátodo deben ir en canales diferentes de la protoboard.
2. Conecte cables puente en uno de los pines digitales del Arduino (para nuestro ejemplo se conecta en el pin digital 7) y realice la conexión positiva con el LED, es decir, en el mismo canal del ánodo.
3. Coloque una resistencia en el mismo canal donde se encuentra el cátodo del LED.
4. Conecte un cable puente a la terminal de tierra del Arduino al canal designado para conexión a tierra (los orificios a la par de la línea azul en la protoboard). Luego otro cable puente hacia el canal donde se encuentra la resistencia.
5. Una vez completado el esquema, conecte el Arduino al computador y coloque el código en el IDE. ¡Cargue el código y listo! El LED debe estar encendiéndose y apagándose cada un segundo en la protoboard.

Código

```
const int led=13; // Define LED como su respectivo pin

void setup(){
  pinMode(LED,OUTPUT); // Define al pin como OUTPUT
}
void loop(){
  digitalWrite(LED,HIGH); // Enciende el LED
}
```

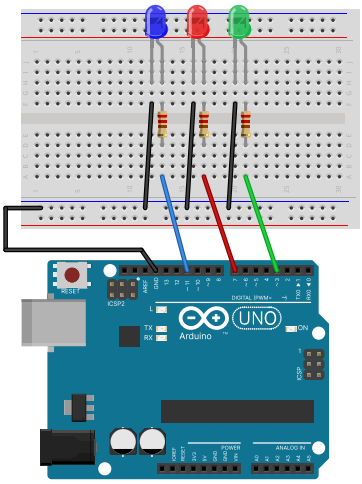
Si la actividad se realiza en grupos, se recomienda hacer cambios, por ejemplo, usar otro pin digital de Arduino (distinto al 13 o 7) y hacer el cambio correspondiente en el código, o cambiar los canales (ya sea vertical u horizontalmente) de la resistencia, el LED, la corriente o tierra, de manera que todos los integrantes sean partícipes. Además, se puede usar la creatividad con el tiempo delay() asignado y la cantidad de LED que se usen, por ejemplo, en la siguiente actividad se creará una pequeña secuencia de luces LED. Comentario

> La función delay() pausa el programa durante el tiempo (en milisegundos) especificado como parámetro. Se sigue que hay 1000 milisegundos en un segundo. Una de las desventajas es que hace que la mayoría de las otras actividades se detengan como lectura de sensores, cálculos o manipulación de pines. Otras funciones similares a delay() son millis() y micros().

iii. Secuencia de luces LED

Siguiendo la misma línea de las actividades anteriores, se procederá a configurar una secuencia de luces LED. Los componentes por utilizar, esquema, y código mantienen la misma idea en términos de conexión y programación. Se deben realizar las conexiones como se observa en el esquema 3.

Esquema 3: secuencia de luces LED



Objetivo:

Secuencia azul, rojo, verde repetidamente.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers wires
4. Luz LED
5. Resistencia 220 Ω

1. Conecte un cable puente a la terminal de tierra del Arduino al canal designado para conexión a tierra (los orificios a la par de la línea azul en la protoboard).
2. Conecte un cable puente en los pines 7, 3 y 11 para hacer la conexión a energía con los tres LED.
3. Conecte una resistencia para cada LED.
4. Conecte cables puentes para hacer la conexión a tierra para cada LED.
5. Conecte un cable puente a la terminal de tierra del Arduino al canal designado para conexión a tierra (los orificios a la par de la línea azul en la protoboard).

Código

```
int pinLed1 = 11; // Define el pin 11 para el primer LED
int pinLed2 = 7;  // Define el pin 7 para el segundo LED
int pinLed3 = 3;  // Define el pin 3 para el tercer LED
int tiempo = 200; // Define el tiempo de espera en milisegundos

void setup() {
  pinMode(pinLed1, OUTPUT); // Configura el pin 11 como salida
  pinMode(pinLed2, OUTPUT); // Configura el pin 7 como salida
  pinMode(pinLed3, OUTPUT); // Configura el pin 3 como salida
      }

      void loop() {
        digitalWrite(pinLed1, HIGH); // Enciende el primer LED
        delay(tiempo);               // Espera el tiempo definido
        digitalWrite(pinLed1, LOW);  // Apaga el primer LED
        delay(tiempo);               // Espera el tiempo definido
        digitalWrite(pinLed2, HIGH); // Enciende el segundo LED
        delay(tiempo);               // Espera el tiempo definido
        digitalWrite(pinLed2, LOW);  // Apaga el segundo LED
        delay(tiempo);               // Espera el tiempo definido
        digitalWrite(pinLed3, HIGH); // Enciende el tercer LED
        delay(tiempo);               // Espera el tiempo definido
        digitalWrite(pinLed3, LOW);  // Apaga el tercer LED
        delay(tiempo);               // Espera el tiempo definido
      }
```

iv. Trabajo independiente

Realice las siguientes tareas según lo aprendido en la sesión:

- ✓ Realice una fila de LED en la que se encienden de izquierda a derecha y viceversa, dando la impresión de un “rebote”. Solución
- ✓ Cree al menos tres secuencias diferentes de forma que el programa las ejecute una después de otra mientras la placa tenga corriente. Solución

Reflexión STEAM

En esta sesión, se introduce a los usuarios a los principios básicos de la electrónica y la programación, dos pilares fundamentales de la ciencia y la tecnología. Al encender y manipular luces LED mediante el uso de una placa de Arduino, los usuarios aprenden sobre circuitos eléctricos, componentes electrónicos (como resistencias y LEDs), y cómo programar microcontroladores para controlar estos componentes. La actividad práctica de escribir y cargar código en la placa de Arduino fortalece la comprensión de la lógica de programación y el flujo de corriente en un circuito.

Las actividades de esta sesión ponen en práctica conceptos de ingeniería electrónica y eléctrica. Los usuarios deben diseñar y ensamblar circuitos simples en un protoboard, realizar conexiones correctas usando jumpers wires, y asegurar que los componentes funcionen según lo esperado. La resolución de problemas es clave aquí, ya que los usuarios deben diagnosticar y corregir errores de conexión o programación para lograr que las luces LED funcionen correctamente.

Las matemáticas están presentes en la necesidad de calcular tiempos y secuencias. Por ejemplo, al establecer un delay() en milisegundos para controlar cuándo se enciende o apaga cada LED, los usuarios aplican conceptos matemáticos de medición de tiempo y secuencias numéricas. Además, el uso de estructuras de control en el código (como loops) requiere una comprensión básica de lógica y estructuras matemáticas.

Sesión 2
Luces LED y botones

1:30 horas	H1-H3-H4-H6

- Reconocer el estado del botón [H1]
- Luz LED con botón [H1]
- Luz LED con dos botones [H1 – H3]
- Luces LED alternadas con botón [H1]
- Trabajo independiente [H1 – H4 – H6]

ESPECIFICACIONES

A partir de ahora, se empezará a usar otros componentes del Starter Kit de Arduino. La complejidad de las siguientes actividades reside en el uso de nuevos componentes, sin embargo, las conexiones que se hacen en la protoboard mantienen la misma estructura como la conexión a energía y tierra, uso de resistencias para controlar la cantidad de energía que necesitan los componentes, y la creatividad para llevar a cabo tareas llamativas.

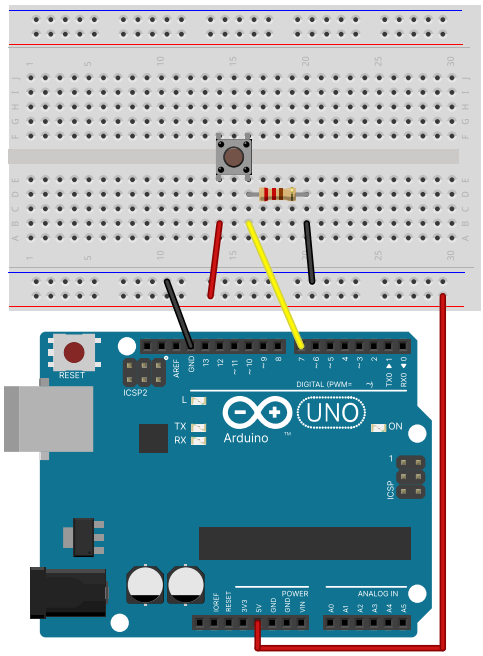
Con esta segunda sesión se trabajarán tanto con las entradas de Arduino como las salidas, es decir, emplearemos inputs/outputs pues con algunos componentes se recibirá información y con otros se enviará información.

i. Reconocer el estado del botón

Dado que ahora se trabajará con inputs, es importante saber cómo leer los datos obtenidos de estos componentes. En este caso, un botón es un componente de entrada (input), por lo que brinda información al Arduino. Es decir, que la placa reconoce cuando este se encuentra presionado o sin presionar de una forma binaria

(1 para presionado y 0 para no presionado). Para esta actividad, se reconocerá cómo y cuándo la placa reconoce estos estados. Para realizarlo, se deben realizar las conexiones como se observa en el esquema 4.

***Esquema 4**: reconocer el estado del botón*



Objetivo:

Reconocer el estado del botón.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers wires
4. Botón
5. Resistencia 220 Ω

1. Conecte un cable puente a la terminal de tierra del Arduino al canal designado para conexión a tierra.
2. Conecte un cable puente en el pin 7 para que el botón pueda “enviar información” a la placa.
3. Conecte una resistencia para el botón y cables puentes para la conexión a energía y tierra.
4. Una vez completado el esquema, conecta el Arduino al computador y coloque el código en el IDE.

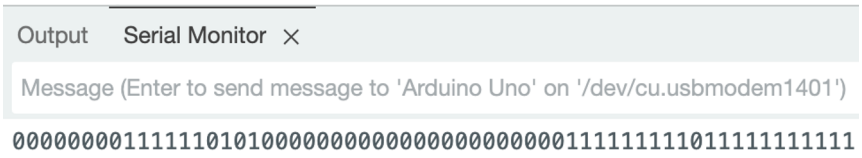
Código

```
1. const int BOTON = 7; // Define el pin 7 para el botón
2. int val; // Variable para almacenar el valor leído del botón
3.
4. void setup() {
5.   pinMode(BOTON, INPUT); // Configura el pin 7 como entrada para el
   botón
6.   Serial.begin(9600); // Inicia la comunicación con el monitor
   serial
7. }
8.
9. void loop() {
10.        val = digitalRead(BOTON); // Lee el estado del botón (HIGH
   o LOW)
11.        Serial.print(val); // Imprime el valor leído del botón en
   el monitor serial
12.        delay(1000); // Espera 1 segundo antes de volver a leer el
   botón
13.      }
```

Una vez ejecutado el código y subido a la placa Arduino, debes ver cómo va apareciendo el número 0 una y otra vez. Si dejas presionado el botón, empezará a salir 1 en vez de 0 durante todo el rato que se mantenga presionado, tal como se observa en la figura 4.

Figura 4:

Estado del botón en el Serial Monitor



La función serial.begin() establece la velocidad de datos en bits por segundo (baudios) para la transmisión de datos en serie. Es importante escribirla en el setup() pues es gracias a esa función que posteriormente podemos usar la función serial.print() que nos permite “imprimir” una variable en el serial monitor.

serial.print() siempre va a imprimir sobre el mismo renglón, tal como se visualizó en la imagen. Si se quiere que imprima hacia abajo, se utiliza la función serial.println()

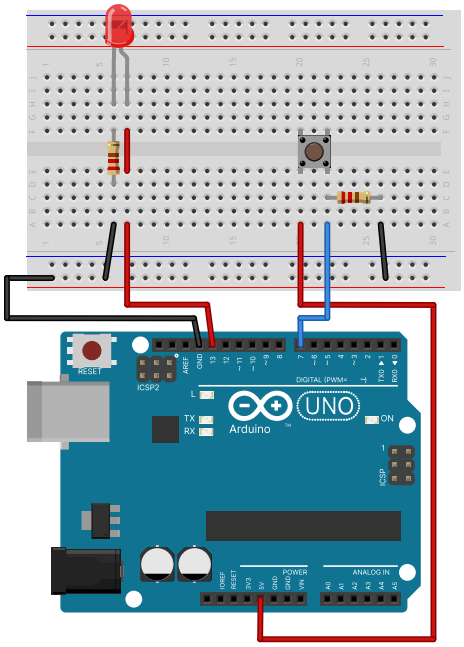
La función digitalRead() lee el estado digital de un pin de entrada (input) y lo devuelve. En el caso de un botón, los estados son:

- HIGH = Presionado
- LOW = No presionado

ii. Luz LED con botón

Ahora, se añadirá un componente nuevo como parte de la actividad. Se utilizará un botón para encender una luz LED al presionarlo, ya sea que se presione rápidamente y la luz de la sensación de parpadear o que se mantenga presionado y el LED se mantenga encendido. Para realizarlo, se deben realizar las conexiones como se observa en el esquema 5.

Esquema 5: luz LED con botón



Objetivo:

Encender y apagar una luz LED al presionar un botón.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers wires
4. Luz LED
5. Resistencia 220 Ω
6. Botón

1. Conecte un cable puente a la terminal de tierra del Arduino al canal designado para conexión a tierra.
2. Conecte un cable puente en el pin 13 para hacer la conexión a energía del LED, y un cable puente en el pin 7 para trasmitir información del botón y otro para hacer la conexión a energía.
3. Conecte una resistencia tanto para el LED como para el botón
4. Conecte cables puentes para hacer la conexión a tierra para el led y el botón.
5. Una vez completado el esquema, conecta el Arduino al computador y coloque el código en el IDE. ¡Cargue el código y listo! Se debe encender el LED al presionar el botón.

Código

```
const int LED = 13;     // Define el pin 13 para el LED
const int BOTON = 7;    // Define el pin 7 para el botón
int val;                // Variable para almacenar el estado del botón

void setup() {
  pinMode(LED, OUTPUT);    // Configura el pin 13 como salida para el LED
  pinMode(BOTON, INPUT);   // Configura el pin 7 como entrada para el botón
}

    void loop() {
      val = digitalRead(BOTON);    // Lee el estado del botón (HIGH o LOW)
      if (val == HIGH) {           // Si el botón está presionado (HIGH)
        digitalWrite(LED, HIGH);   // Enciende el LED
      } else {                      // Si el botón no está presionado (LOW)
        digitalWrite(LED, LOW);    // Apaga el LED
      }
    }
```

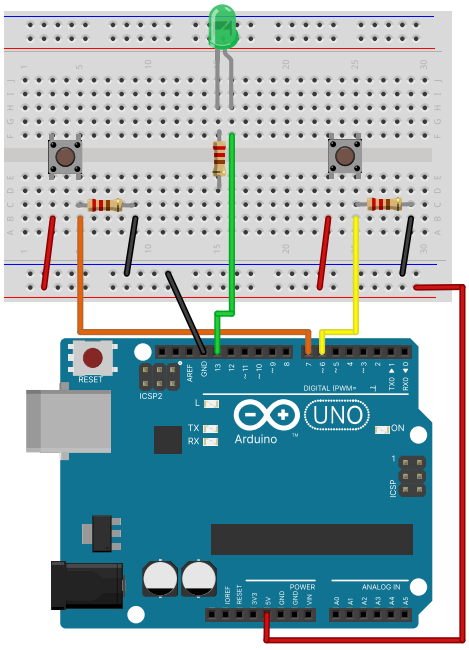
Poco a poco se irá aprendiendo nuevas funciones que podemos usar en el código, en esta última actividad utilizamos la función condicional if. Por ahora, se ha

logrado hacer que un LED se encienda cuando el botón esté presionado. Ahora se programará que al presionar una vez un botón, el LED quede encendido y al presionar otro botón, el LED se apague.

iii. Luz LED con dos botones

Ahora, se combinarán componentes de los que ya se sabe su funcionamiento. Los componentes por utilizar, esquema, y código mantienen la misma idea en términos de conexión y programación. Para realizarlo, se deben realizar las conexiones como se observa en el esquema 6.

Esquema 6: luz LED con dos botones



Objetivo:

Encender un LED al presionar un botón y apagarlo al presionar otro.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers wires
4. Luz LED
5. Resistencia 220 Ω
6. Botón

1. Conecte un cable puente a la terminal de tierra del Arduino al canal designado para conexión a tierra.
2. Conecte un cable puente en el pin 13 para hacer la conexión a energía del LED, y un cable puente en el pin 7 para para trasmitir información del botón y otro para hacer la conexión a energía.
3. Conecte una resistencia tanto para el LED como para el botón.
4. Conecte cables puentes para hacer la conexión a tierra para el led y el botón.
5. Una vez completado el esquema, conecta el Arduino al computador y coloque el código en el IDE. ¡Carga el código y listo! Debes poder encender el LED al presionar el botón.

Código

```
const int LED = 13;       // Define el pin 13 para el LED
const int BOTON_E = 7;    // Define el pin 7 para el botón de encendido
const int BOTON_A = 6;    // Define el pin 6 para el botón de apagado
int val_E;                // Variable para almacenar el estado del botón de encendido
int val_A;                // Variable para almacenar el estado del botón de apagado

void setup() {
  pinMode(LED, OUTPUT);    // Configura el pin 13 como salida para el LED
  pinMode(BOTON_A, INPUT); // Configura el pin 6 como entrada para el botón de apagado
        pinMode(BOTON_E, INPUT); // Configura el pin 7 como entrada para el botón de encendido
      }

      void loop() {
        val_E = digitalRead(BOTON_E);  // Lee el estado del botón de encendido (HIGH o LOW)
        val_A = digitalRead(BOTON_A);  // Lee el estado del botón de apagado (HIGH o LOW)
        if (val_E == HIGH) {           // Si se presiona el botón de encendido
          digitalWrite(LED, HIGH);     // Enciende el LED
        }
```

```
19.         if (val_A == HIGH) {           // Si se presiona el botón
   de apagado
20.           digitalWrite(LED, LOW);      // Apaga el LED
21.         }
22.       }
```

Al usar la creatividad se pueden crear otro tipo de actividades con las cuales entretenerse, asombrarse y reforzar conocimiento ¡solo usando luces LED! Mientras más LED, más secuencias se pueden crear.

iv. Trabajo independiente

Realice las siguientes tareas según lo aprendido en la sesión:

- ✓ Configure una secuencia de luces LED y cambie la dirección utilizando dos botones. Solución
- ✓ Programe tres secuencias de luces LED y con la ayuda de un botón pueda cambiar de una a otra. Solución

Reflexión STEAM

En esta sesión, se amplía la comprensión de los usuarios sobre la ciencia y la tecnología al introducir componentes como botones y LEDs, y trabajar con entradas y salidas de Arduino. Reconocer el estado del botón y utilizarlo para controlar LEDs implica entender cómo funcionan los sensores y actuadores, así como los principios de la comunicación digital entre el hardware y el software. Los usuarios aprenden a leer datos de entrada (input) y a programar respuestas de salida (output), fortaleciendo su conocimiento en electrónica y programación.

La sesión se centra en diseñar y ensamblar circuitos que combinan botones y LEDs. Los usuarios deben realizar conexiones precisas en la protoboard y asegurarse de que los componentes estén configurados correctamente para funcionar según lo esperado. La actividad de encender y apagar LEDs con uno o dos botones fomenta el desarrollo de habilidades en ingeniería eléctrica y la resolución de problemas técnicos. La creatividad se fomenta mediante la programación de secuencias de luces LED y diferentes configuraciones con los botones. Los usuarios pueden experimentar con diferentes patrones de encendido y apagado, ajustando los tiempos de delay() y el número de LEDs utilizados. Este aspecto artístico permite a los usuarios expresar su creatividad y desarrollar una sensibilidad estética mientras trabajan en un contexto técnico.

Las actividades requieren aplicar conceptos matemáticos básicos, como la lógica binaria (1 para presionado y 0 para no presionado) y el uso de estructuras de control condicional en el código. Al programar el encendido y apagado de LEDs con botones, los usuarios utilizan operaciones lógicas y condicionales, que son fundamentales en la programación y la lógica matemática. Estas actividades refuerzan el entendimiento de cómo las matemáticas se integran en la programación y la electrónica.

Sesión 3

Matriz 8x8

1:30 horas	H1-H2-H4-H6

- Encender cada luz de la matriz a mano [H1 – H4]
- Recorrer la matriz [H1]
- Imprimir la silueta de un gato [H2 – H6]
- Escribir una frase [H1]
- Trabajo independiente [H1 – H2 – H6]

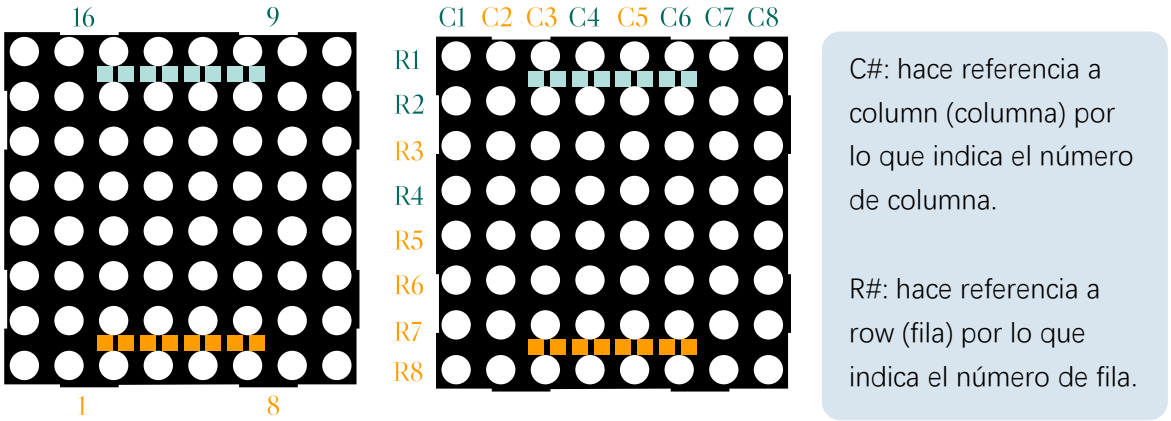
ESPECIFICACIONES

El componente por utilizar en esta sesión será la Matriz 8x8. Existen varias versiones con diferente pin de conexión, por lo que es importante recalcar que se utilizará la matriz 1588bs que contiene 64 leds y 16 pines. Con este tipo de matriz, podremos imprimir cualquier silueta mensaje que queramos.

A pesar de que una matriz de luces LED es bastante útil, es uno de los componentes más tediosos de conectar en la protoboard, pues la forma de encender cada LED de manera independiente requiere de un análisis previo. Primero, se debe conocer a qué hacer referencia cada uno de los pines de la matriz, esto se puede observar el la figura 5.

Figura 5

Pines y referencia de filas-columnas de la matriz 8x8



(a) Pines del 1 al 16 (b) Relación C#-R#

En la figura 5 se observa en (a) la ubicación de los pines y en (b) se tiene una referencia de cuáles serían las filas y columnas en la matriz. La relación que existe entre ellas se explicita en la tabla 5. Por ejemplo, con el pin 3 se configura la C2, con el pin 12 se programa el R4, y así respectivamente con los demás pines.

Tabla 5

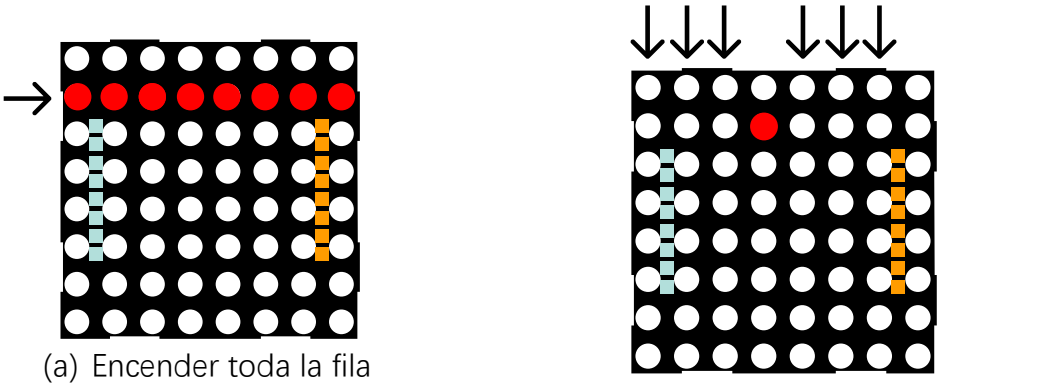
Relación pines con las columnas y filas de la matriz 8x8

Pin	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Matriz	R5	R7	C2	C3	R8	C5	R6	R3	R1	C4	C6	R4	C1	R2	C7	C8

Tomando como ejemplo la figura 6, se muestra que para encender un LED en particular se debe encender una fila y apagar el resto de las columnas (se enciende R2 y se apagan las columnas C1, C2, C3, C5, C6, C7, C8).

Figura 6

Encender un LED particular de la matriz 8x8



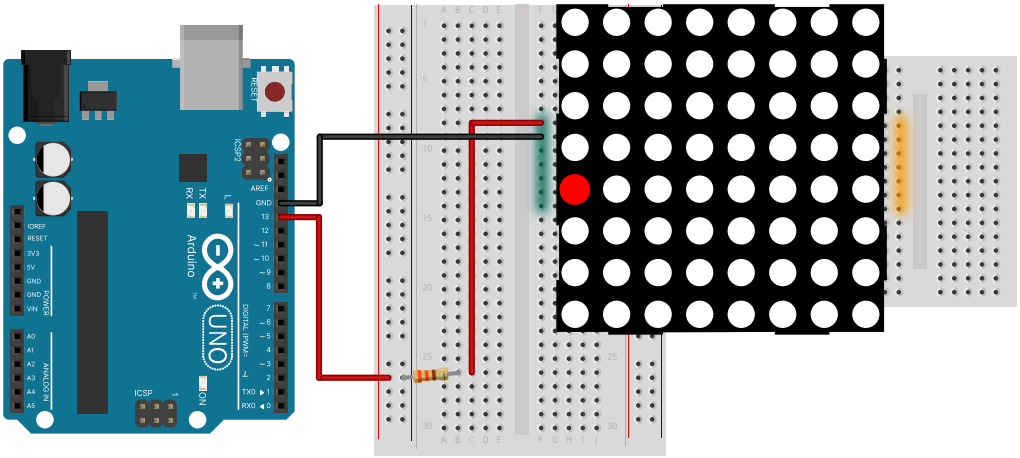
(a) Encender toda la fila (b) Apagar el resto de las columnas

Ahora, se conectará la matriz al Arduino. Es usual que se necesite una extensión de la protoboard para poder conectar la matriz. Dependiendo del modelo, se puede saber dónde se ubican los pines mencionados anteriormente. En este caso, en el lado donde se encuentra el nombre impreso se ubican los pines del 1 al 8. Antes de hacer todas las conexiones se intentará encender cada luz LED primero y así saber que cada pin está bien ubicado.

i. Encender cada LED de la matriz a mano

Siguiendo el esquema 7 a continuación, se cambiarán de posición los cables puente para ir encendiendo cada led.

Esquema 7: encender cada LED de la matriz a mano



Objetivo:

Encender cada LED a mano.

Materiales:

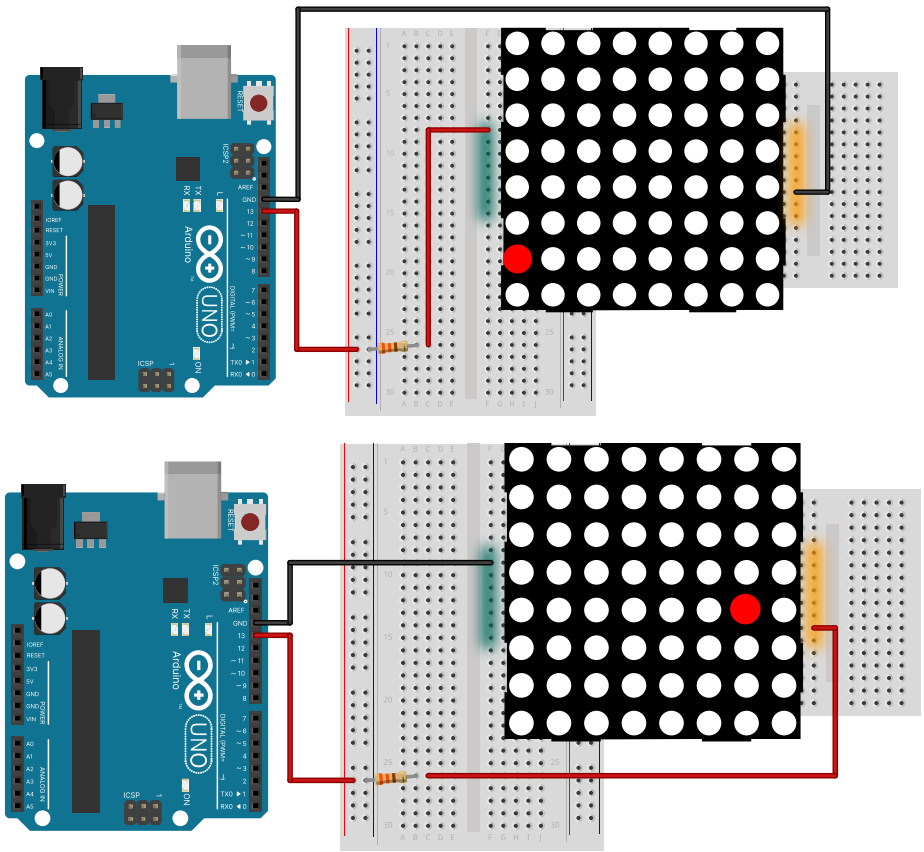
1. Placa de Arduino
2. Protoboard
3. Jumpers Wires
4. Resistencia 330 Ω
5. Matriz 8x8

1. Conecte un cable puente a uno de los pines de la matriz y realice la conexión a tierra.
2. Conecte un cable puente en el pin 13 de la placa de Arduino para que comparta conexión con otro de los pines de la matriz, recuerde usar la resistencia.

Se utilizará el jumper wire que está alineado con la resistencia para las columnas y el jumper wire alineado a tierra para las filas, esto para evitar quemar los LED. Si se mantiene el jumper wire en C1 y se mueve el otro hacia otros pines de fila se pueden encender otros LED de la misma columna, de la misma forma si se decide cambiar las columnas. Un ejemplo de lo anterior se muestra en la figura 7.

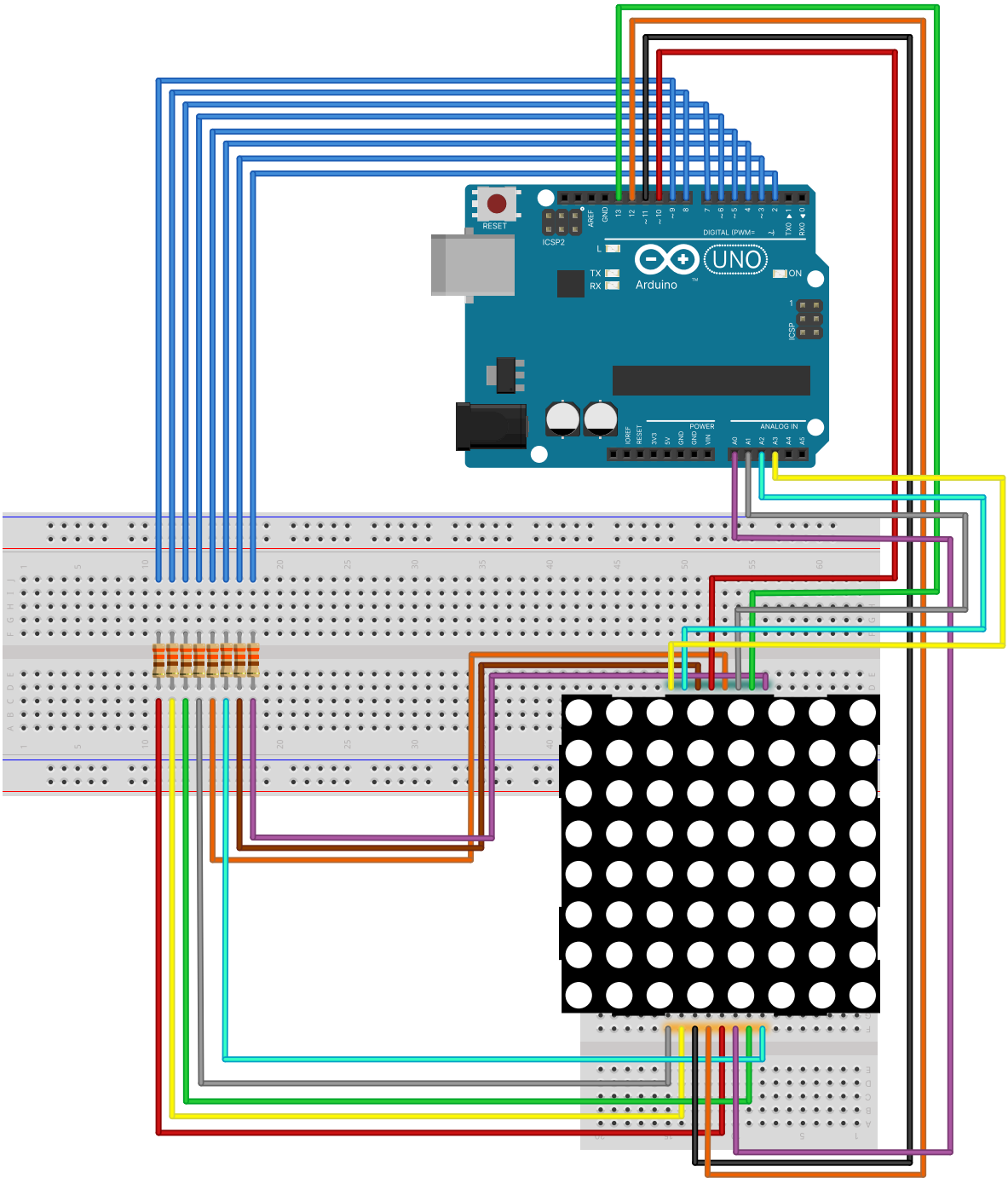
Figura7

Ejemplos para encender un LED particular de la matriz 8x8



Si se encendió otro LED, significa que debes dar vuelta a la matriz. Una vez revisado y verificado que todos los LEDs funcionan correctamente, se procederá con las actividades, para ello se conectará la matriz completa. Se puede observar que conectar una matriz a Arduino se vuelve muy tedioso. El lado positivo de esta conexión es que se usará para las actividades que restan de la sesión.

Esquema # 8: conexión eléctrica de la matriz 8x8



ii. Recorrer la matriz

Esta actividad tiene como objetivo encender cada luz LED de la matriz una vez se ejecute el código. Existen varias librerías que permiten un uso más sencillo e intuitivo

de la matriz; sin embargo, esta sesión se trabajará sin ninguna de ellas para conocer a fondo el funcionamiento de la matriz.

Código

```
1. // Definición de los pines para las filas (R1-R8) y columnas (C1-
   C8) de la matriz LED
2. #define R1 2
3. #define R2 3
4. #define R3 4
5. #define R4 5
6. #define R5 6
7. #define R6 7
8. #define R7 8
9. #define R8 9
10.      #define C1 10
11.      #define C2 11
12.      #define C3 12
13.      #define C4 13
14.      #define C5 A0
15.      #define C6 A1
16.      #define C7 A2
17.      #define C8 A3
18.
19.      void setup() {
20.        // Configuración de los pines como salida para las filas y
   columnas de la matriz LED
21.        pinMode(R1, OUTPUT);
22.        pinMode(R2, OUTPUT);
23.        pinMode(R3, OUTPUT);
24.        pinMode(R4, OUTPUT);
25.        pinMode(R5, OUTPUT);
26.        pinMode(R6, OUTPUT);
27.        pinMode(R7, OUTPUT);
28.        pinMode(R8, OUTPUT);
29.        pinMode(C1, OUTPUT);
30.        pinMode(C2, OUTPUT);
31.        pinMode(C3, OUTPUT);
32.        pinMode(C4, OUTPUT);
33.        pinMode(C5, OUTPUT);
34.        pinMode(C6, OUTPUT);
35.        pinMode(C7, OUTPUT);
36.        pinMode(C8, OUTPUT);

37.        // Inicialización de las columnas en estado alto (apagadas)
38.        digitalWrite(C1, HIGH);
39.        digitalWrite(C2, HIGH);
```

```
40.         digitalWrite(C3, HIGH);
41.         digitalWrite(C4, HIGH);
42.         digitalWrite(C5, HIGH);
43.         digitalWrite(C6, HIGH);
44.         digitalWrite(C7, HIGH);
45.         digitalWrite(C8, HIGH);
46.       }
47.
48.       void loop() {
49.         // Bucle para encender cada fila secuencialmente
50.         for (int fila = 0; fila < 8; fila++) {
51.           digitalWrite(R1 + fila, HIGH); // Enciende la fila
    correspondiente
52.         // Bucle para encender cada columna secuencialmente
53.           for (int columna = 0; columna < 8; columna++) {
54.             digitalWrite(C1 + columna, LOW); // Enciende la columna
    correspondiente
55.             delay(100); // Espera un tiempo corto
56.             digitalWrite(C1 + columna, HIGH); // Apaga la columna
    correspondiente
57.           }
58.           delay(2); // Espera un breve período entre filas
59.           // Apaga todas las filas después de mostrar una fila
60.           digitalWrite(R1, LOW);
61.           digitalWrite(R2, LOW);
62.           digitalWrite(R3, LOW);
63.           digitalWrite(R4, LOW);
64.           digitalWrite(R5, LOW);
65.           digitalWrite(R6, LOW);
66.           digitalWrite(R7, LOW);
67.           digitalWrite(R8, LOW);
68.         }
69.       }
```

Al ejecutar el código se debe observar un “recorrido” por todas las luces LED, siguiendo un patrón de izquierda a derecha y luego fila por fila. El código funciona de la siguiente manera: como se observó anteriormente, para encender un solo LED de la matriz es necesario encender toda la fila y luego apagar el resto de las columnas. Para ello se utilizan los ciclos for (), el primer for () determina una fila y no la va a cambiar hasta que termine el for () de adentro; el segundo for (), se encarga de apagar el resto de las columnas.

iii. Imprimir la silueta de un gato

El objetivo de esta actividad es proyectar un gato en la matriz. La matriz permite en gran medida explotar la creatividad, una vez comprendido el funcionamiento de los pines y su programación se pueden generar diversas figuras o representaciones.

Código

```
// Definición de los pines para las filas (R1-R8) y columnas (C1-C8) de la matriz LED
#define R1 2
#define R2 3
#define R3 4
#define R4 5
#define R5 6
#define R6 7
#define R7 8
#define R8 9
      #define C1 10
      #define C2 11
      #define C3 12
      #define C4 13
      #define C5 A0
      #define C6 A1
      #define C7 A2
      #define C8 A3

      // Definición de la matriz de 8x8 para representar el estado de cada LED (0 = apagado, 1 = encendido)
      int Matriz[8][8] = {
        {0, 1, 1, 1, 0, 1, 1, 1},
        {0, 0, 0, 0, 0, 1, 1, 1},
        {0, 1, 0, 1, 0, 1, 1, 1},
        {1, 0, 0, 0, 1, 1, 1, 0},
        {1, 1, 0, 1, 1, 1, 1, 0},
        {1, 0, 0, 0, 0, 1, 1, 0},
        {1, 0, 0, 0, 0, 0, 1, 0},
        {0, 1, 0, 0, 0, 0, 0, 1},
      };

      void setup() {
      // Configuración de los pines como salida para las filas y columnas de la matriz LED
        pinMode(R1, OUTPUT);
        pinMode(R2, OUTPUT);
        pinMode(R3, OUTPUT);
        pinMode(R4, OUTPUT);
```

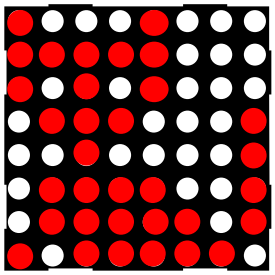
```
37.         pinMode(R5, OUTPUT);
38.         pinMode(R6, OUTPUT);
39.         pinMode(R7, OUTPUT);
40.         pinMode(R8, OUTPUT);
41.         pinMode(C1, OUTPUT);
42.         pinMode(C2, OUTPUT);
43.         pinMode(C3, OUTPUT);
44.         pinMode(C4, OUTPUT);
45.         pinMode(C5, OUTPUT);
46.         pinMode(C6, OUTPUT);
47.         pinMode(C7, OUTPUT);
48.         pinMode(C8, OUTPUT);
49.       }
50.
51.       void loop() {
52.         // Bucle para encender cada fila secuencialmente
53.         for (int fila = 0; fila < 8; fila++) {
54.           digitalWrite(R1 + fila, HIGH); // Enciende la fila
    correspondiente
55.         // Configura cada columna según el valor en la matriz (0 o
    1)
56.           digitalWrite(C1, Matriz[fila][0]);
57.           digitalWrite(C2, Matriz[fila][1]);
58.           digitalWrite(C3, Matriz[fila][2]);
59.           digitalWrite(C4, Matriz[fila][3]);
60.           digitalWrite(C5, Matriz[fila][4]);
61.           digitalWrite(C6, Matriz[fila][5]);
62.           digitalWrite(C7, Matriz[fila][6]);
63.           digitalWrite(C8, Matriz[fila][7]);
64.           delay(2); // Espera un breve período antes de pasar a la
    siguiente fila
65.           // Apaga todas las filas después de mostrar una fila
66.           digitalWrite(R1, LOW);
67.           digitalWrite(R2, LOW);
68.           digitalWrite(R3, LOW);
69.           digitalWrite(R4, LOW);
70.           digitalWrite(R5, LOW);
71.           digitalWrite(R6, LOW);
72.           digitalWrite(R7, LOW);
73.           digitalWrite(R8, LOW);
74.         }
75.       }
```

La interacción con este código es mayor, pues como se nota, al inicio se define un arreglo de ocho listas que se puede asemejar a una matriz 8x8, donde 1 significa encendido y 0 significa apagado. Esta relación se observa en la figura 8.

Figura 8

Representación de gato en la matriz con 1 y 0

```
INT MATRIZ [8][8] ={
  {0,1,1,1,0,1,1,1},
  {0,0,0,0,0,0,1,1},
  {0,1,0,1,0,1,1,1},
  {1,0,0,0,1,1,1,0},
  {1,1,0,1,1,1,1,0},
  {1,0,0,0,0,0,1,0},
  {1,0,0,0,0,0,0,0},
  {0,1,0,0,0,0,0,1},
};
```



Siguiendo la idea anterior, se podrían mostrar otras figuras como caritas felices, tristes, representaciones de aprobación y desaprobación, entre otros. Así como se puede dibujar cualquier objeto que pueda configurarse en la matriz 8x8, también se puede escribir. Por ello, en esta última actividad, se escribirá una frase y que esta se vaya trasladando hacia la izquierda.

iv. Escribir una frase

El objetivo de esta actividad es mostrar una frase que se vaya trasladando de izquierda a derecha. En las siguientes páginas se muestra el código.

Código

```
1. // Definición de los pines para las filas (R1-R8) y columnas (C1-
   C8) de la matriz LED
2. #define R1 2
3. #define R2 3
4. #define R3 4
5. #define R4 5
6. #define R5 6
7. #define R6 7
8. #define R7 8
9. #define R8 9
10.      #define C1 10
11.      #define C2 11
12.      #define C3 12
13.      #define C4 13
14.      #define C5 A0
15.      #define C6 A1
```

```
16.      #define C7 A2
17.      #define C8 A3
18.      int position = 0; // Inicializa la posición del
   desplazamiento
19.      // Definición de la matriz que representa el mensaje "Hola
   gente" en un tamaño de 8x54
20.      int Hola_gente[8][54] = {
21.      {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
   1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
   1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
22.      {1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0,
   0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0,
   0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1},
23.      {1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1,
   0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1,
   0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1},
24.      {1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0,
   0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1,
   0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1},
25.      {1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1,
   0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,
   0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1},
26.      {1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1,
   0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,
   0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1},
27.      {1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1,
   0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1,
   0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1},
28.      {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
   1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
   1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
29.      };
30.      void setup() {
31.        // Configuración de los pines como salida para las filas y
   columnas de la matriz LED
32.        pinMode(R1, OUTPUT);
33.        pinMode(R2, OUTPUT);
34.        pinMode(R3, OUTPUT);
35.        pinMode(R4, OUTPUT);
36.        pinMode(R5, OUTPUT);
37.        pinMode(R6, OUTPUT);
38.        pinMode(R7, OUTPUT);
39.        pinMode(R8, OUTPUT);
40.        pinMode(C1, OUTPUT);
41.        pinMode(C2, OUTPUT);
42.        pinMode(C3, OUTPUT);
43.        pinMode(C4, OUTPUT);
44.        pinMode(C5, OUTPUT);
```

```
45.         pinMode(C6, OUTPUT);
46.         pinMode(C7, OUTPUT);
47.         pinMode(C8, OUTPUT);
48.       }
49.       void loop() {
50.         for (int x = 0; x < 5; x++) { // Repite 5 veces el ciclo
    para mostrar el mensaje
51.           for (int fila = 0; fila < 8; fila++) {
52.             digitalWrite(R1 + fila, HIGH); // Enciende la fila
    correspondiente
53.             // Configura cada columna según el valor en la matriz
    desplazada por 'position'
54.             digitalWrite(C1, Hola_gente[fila][(0 + position) %
    54]);
55.             digitalWrite(C2, Hola_gente[fila][(1 + position) %
    54]);
56.             digitalWrite(C3, Hola_gente[fila][(2 + position) %
    54]);
57.             digitalWrite(C4, Hola_gente[fila][(3 + position) %
    54]);
58.             digitalWrite(C5, Hola_gente[fila][(4 + position) %
    54]);
59.             digitalWrite(C6, Hola_gente[fila][(5 + position) %
    54]);
60.             digitalWrite(C7, Hola_gente[fila][(6 + position) %
    54]);
61.             digitalWrite(C8, Hola_gente[fila][(7 + position) %
    54]);
62.             delay(2); // Espera un breve período antes de pasar a
    la siguiente fila
63.             // Apaga todas las filas después de mostrar una fila
64.             digitalWrite(R1, LOW);
65.             digitalWrite(R2, LOW);
66.             digitalWrite(R3, LOW);
67.             digitalWrite(R4, LOW);
68.             digitalWrite(R5, LOW);
69.             digitalWrite(R6, LOW);
70.             digitalWrite(R7, LOW);
71.             digitalWrite(R8, LOW);
72.           }
73.         }
74.         position = position + 1; // Incrementa la posición para
    desplazar el mensaje
75.       }
```

El código mostrado anteriormente funciona muy similar al anterior, la única diferencia es que el arreglo es de 8x54. La razón de esto es porque el código lo que

hace es deslizarse a través de ese arreglo de forma cíclica, como si el texto estuviera girando sobre la matriz indefinidamente. Si se estirara el arreglo se puede apreciar mejor como se muestra en la figura 9.

Figura 9

Representación del texto "hola gente" en la matriz con 1 y 0

v. Trabajo independiente

Realice las siguientes tareas según lo aprendido en la sesión:

- ✓ Recorra todos los LEDs en sentido opuesto al visto en la sesión. Solución
- ✓ Dibuje una carita feliz en la matriz o cualquier otro objeto de su preferencia. Solución
- ✓ Escriba una frase e imprímela en la matriz. Solución

Reflexión STEAM

En esta sesión, la ciencia y la tecnología están presentes en el análisis y comprensión de cómo funciona la matriz LED, así como en la programación del Arduino para controlarla. se necesita habilidad tecnológica para programar el Arduino y utilizar los pines de entrada y salida de manera eficiente.

La ingeniería se manifiesta en la conexión física de la matriz LED al Arduino y en la resolución de problemas relacionados con la configuración y el funcionamiento de la matriz. Esto incluye la identificación de pines, el diseño de circuitos y la solución de problemas durante la conexión y programación.

El aspecto artístico se encuentra en la creatividad para diseñar y representar formas visuales en la matriz LED. Por ejemplo, la actividad de dibujar una carita feliz implica utilizar la matriz como lienzo para expresar una forma artística y que se evidencia también con la representación de figuras como el gato o la impresión de frases.

Las matemáticas están presentes en varios aspectos, como la organización y manipulación de datos en forma de matrices, la comprensión de coordenadas para encender LEDs específicos en la matriz, y el cálculo de tiempos y secuencias en la programación del Arduino. Además, en la actividad de desplazamiento de la frase, se utiliza la lógica matemática para controlar la posición y el movimiento de los elementos en la matriz.

Sesión 4

Funcionamiento de display LCD

1:00 hora | H1-H2-H3-H4-H5-H6

- Hola mundo y contador de segundos [H1]
- Texto deslizante [H1]
- Display, botón y LED [H1 – H4]
- Trabajo independiente [H1 – H4 – H6]

ESPECIFICACIONES

Ahora se conocerá uno de los componentes más usados en Arduino, el LCD. La Pantalla de Cristal Líquido (LCD, por sus siglas en inglés) tiene una interfaz paralela, lo que significa que la placa tiene que manipular varios pines de la interfaz a la vez para controlar la pantalla, más adelante se estudiarán estos pines.

El tamaño de un LCD depende de la cantidad de áreas rectangulares ubicadas, para esa unidad se utilizará un LCD de 16x2. Además, este tipo de LCD cuenta con un total de 16 pines distribuidos como se muestra en la figura 10 y se describen en la tabla 6.

Figura 10

Pines en el LCD

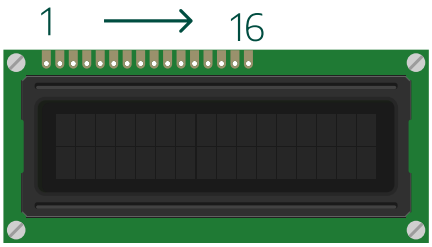


Tabla 6

Descripción de los pines de la pantalla LCD

Pin	Descripción
PIN 1. VSS	La “S” hace referencia a “serie”, significa que es una conexión común, por lo que es la conexión al terminal de tierra.
PIN 2. VDD	La “D” hace referencia a “dispositivo”, por lo que se conecta al voltaje que requiere el dispositivo (5 voltios).
PIN 3. V0	Controla el contraste del LCD, por lo que va conectado al potenciómetro.
PIN 4. RS	Es el pin que registra y selecciona las instrucciones al LCD.
PIN 5. RW	Hace referencia a read/write (leer y escribir) y determina cómo se va a leer y escribir sobre el LDC, usualmente se usan los ya establecidos por el componente.
PIN 6. E	Se refiere a enable (permitir) por lo que permite escribir en los registros.
PIN 7-14. D0-D7	Con estos 8 pines se puede escribir en el LCD pues usa código ASCII, por ejemplo 0100 0001 imprime una “A”; sin embargo, con el uso de librerías se puede reducir a usar solo 4 de estos pines.
PIN 15. A	Es el ánodo de la luz trasera del LCD.
PIN 16. K	Es el cátodo de la luz trasera del LCD.

Ahora que ya se abarcó el funcionamiento básico de los pines del LCD es momento de iniciar con las actividades.

i. Hola mundo y contador de minutos

En el esquema 9 se muestra las conexiones eléctricas para la pantalla LCD. En la figura 11 se señalan los pines del potenciómetro, este se puede regular al girar la pieza inferior.

Esquema 9: hola mundo y contador de minutos

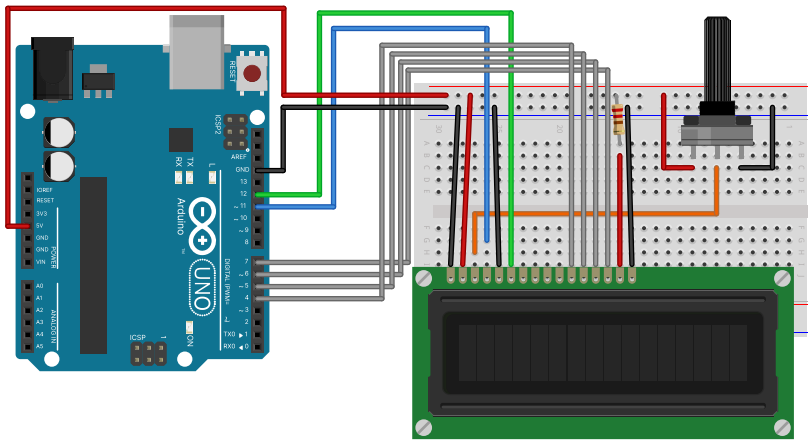
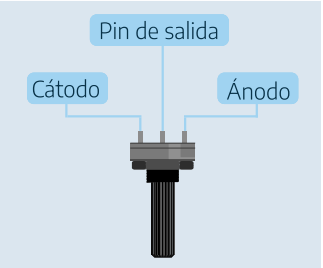


Figura 11
Pines del potenciómetro



Objetivo:

Imprimir Hola Mundo y
un cronómetro de segundos.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers Wires
4. Resistencia 220 Ω
5. 16x2 LCD
6. Potenciómetro

Se debe instalar la librería *LiquidCrystal*, para ello basta buscarlo en la opción de librerías y descargarlo.

Si al iniciar el programa, solo es enciende el LCD, pero no se lee nada, debe ajustar el potenciómetro para que haya un contraste justo que permita la lectura de los datos. En la siguiente actividad, se logrará que el texto "rebote" en el LCD. Para ello, las conexiones del LCD no cambian.

1. Conecte un cable puente a la terminal de tierra del Arduino al canal designado para conexión a tierra.
2. Conecte un cable puente en el pin 5V del Arduino al canal designado para conexión a energía.
3. Conecte el pin 1 y pin 16 al canal a tierra.
4. Conecte el pin 2 y pin 5 al canal de energía.
5. Conecte el pin 4 al pin de salida del potenciómetro.
6. Conecte el pin 3 y pin 6 al pin 11 y pin 12 del Arduino respectivamente
7. Conecte los pines 11, 12, 13 y 14 a los pines 4, 5, 6 y 7 del Arduino respectivamente.
8. Conecte el pin 15 a la resistencia que está conectada al canal de energía
9. Conecte el ánodo y cátodo del potenciómetro.
10. Una vez completado el esquema, se conecta el Arduino al computador y se coloca el código en el IDE.

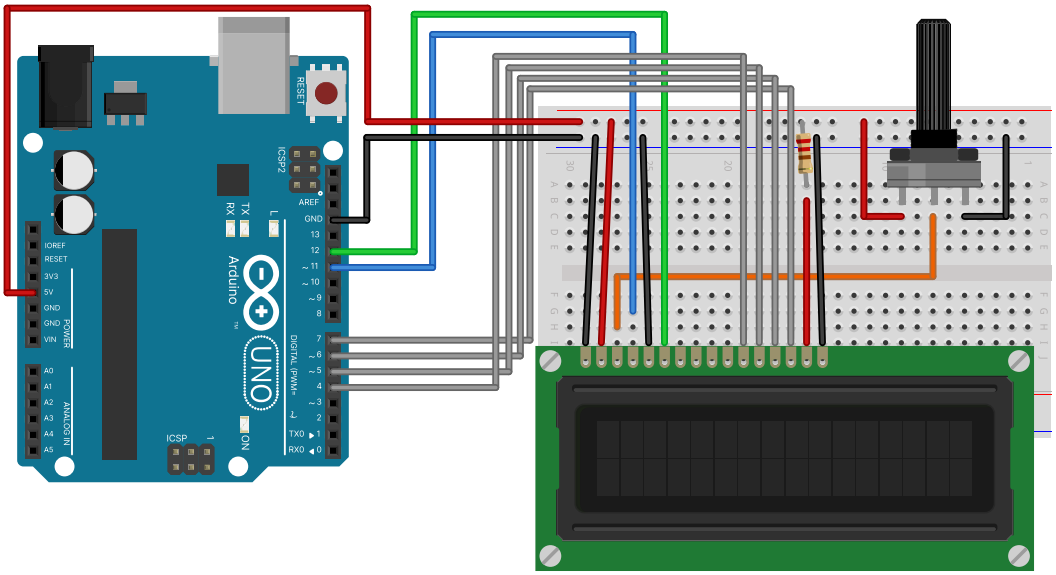
Código

```
#include <LiquidCrystal.h> // Incluye la librería LiquidCrystal para controlar el LCD
const int rs = 11, en = 12, d4 = 4, d5 = 5, d6 = 6, d7 = 7; // Define los pines para la conexión del LCD
LiquidCrystal lcd(rs, en, d4, d5, d6, d7); // Inicializa el objeto lcd con los pines definidos
void setup() {
lcd.begin(16, 2); // Configura el LCD con 16 columnas y 2 filas
lcd.print("Hola, Mundo!"); // Imprime el mensaje "Hola, Mundo!" en la primera línea del LCD
}
void loop() {
lcd.setCursor(0, 1); // Coloca el cursor en la primera columna de la segunda fila
      lcd.print(millis() / 1000); // Imprime el número de segundos transcurridos desde que el Arduino comenzó a ejecutar el programa
      }
```

ii. Texto deslizante

En el esquema 10 se muestra las conexiones eléctricas para la pantalla LCD.

Esquema 10: texto deslizante



Nota: elaboración propia

Objetivo:

Imprimir un texto que vaya de un lado del LCD al otro.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers Wires
4. Resistencia 220 Ω
5. 16x2 LCD
6. Potenciómetro

Los pasos por seguir para hacer la conexión corresponden a los mismos de la actividad anterior. El deslizamiento del texto va a depender del largo de la frase, eso quiere decir que se moverá a la izquierda primero la cantidad de veces según la longitud del texto. Luego a la derecha la misma cantidad más la longitud del LCD para que llegue al otro extremo y desaparezca, por último, se mueve a la izquierda la misma cantidad para volver al inicio.

Código

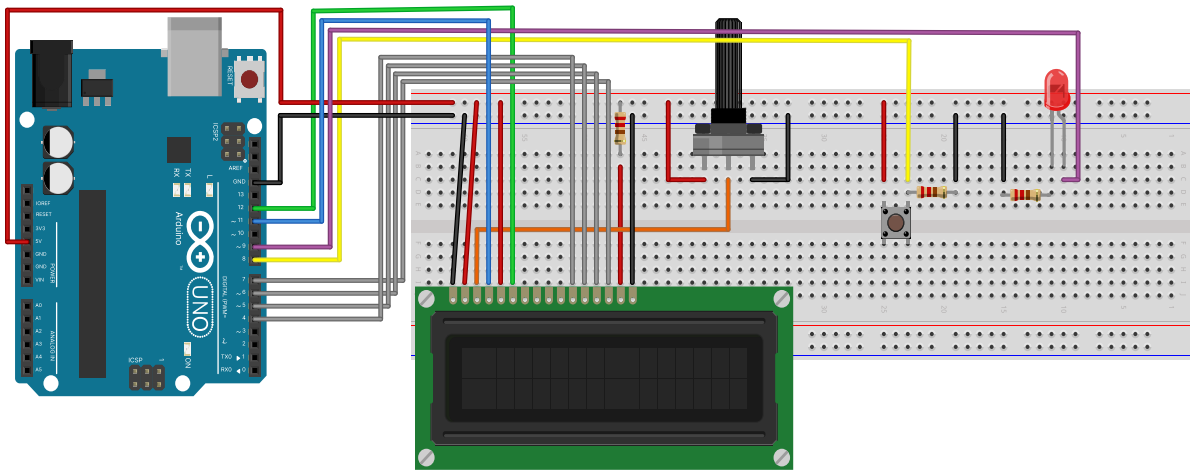
```
#include <LiquidCrystal.h> // Incluye la librería LiquidCrystal para controlar el LCD
const int rs = 11, // Define el pin para rs
en = 12, // Define el pin para en
d4 = 4,  // Define el pin para d4
d5 = 5,  // Define el pin para d5
d6 = 6,  // Define el pin para d6
d7 = 7;  // Define el pin para d7
LiquidCrystal lcd(rs, en, d4, d5, d6, d7); // Inicializa el objeto lcd con los pines definidos
String Texto = "Deslizar"; // Define el texto que se va a mostrar y desplazar en el LCD

      void setup() {
        lcd.begin(16, 2); // Configura el LCD con 16 columnas y 2 filas
        lcd.print(Texto); // Imprime el texto en la primera línea del LCD
        delay(2000); // Espera 2 segundos
      }

      void loop() {
        for (int positionCounter = 0; positionCounter < Texto.length(); positionCounter++) {
          lcd.scrollDisplayLeft(); // Desplaza el texto hacia la izquierda
          delay(500); // Espera 0.5 segundos
        }
        for (int positionCounter = 0; positionCounter < (Texto.length() + 16); positionCounter++) {
          lcd.scrollDisplayRight(); // Desplaza el texto
          delay(500); // Espera 0.5 segundos
        }
        for (int positionCounter = 0; positionCounter < (Texto.length() + 16); positionCounter++) {
          lcd.scrollDisplayLeft(); // Desplaza el texto
          delay(500); // Espera 0.5 segundos
        }
        delay(2000); // Espera 2 segundos antes de repetir el ciclo
      }
```

El efecto deslizante de la actividad anterior se logra por la longitud de la frase inicial “Deslizar”, cambie la frase y el código de manera que no se pierda el efecto.

iii. Display, botón y LED

En el esquema 11 se visualiza las conexiones eléctricas para esta actividad.

***Esquema 11**: display, botón y LED*



Objetivo:

Mostrar en la pantalla LCD
el estado del LED encendido-apagado.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers Wires
4. Resistencia 220 Ω
5. 16x2 LCD
6. Potenciómetro

Con el código reciente, se consiguió la capacidad de reflejar o "imprimir" lo que ocurre con otros componentes en la pantalla LCD. Esta funcionalidad resulta especialmente útil para verificar los sensores y otros dispositivos.

Código

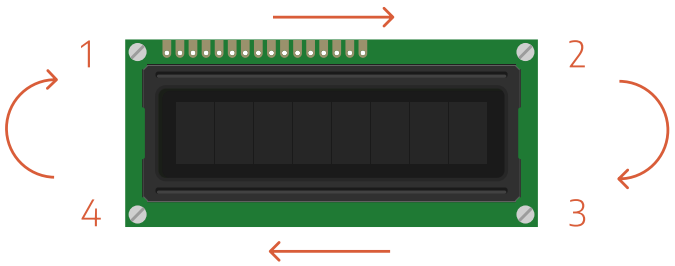
```
1. #include <LiquidCrystal.h> // Incluye la librería LiquidCrystal
   para controlar el LCD
2. const int rs = 11, en = 12, d4 = 4, d5 = 5, d6 = 6, d7 = 7; //
   Define los pines para la conexión del LCD
3. LiquidCrystal lcd(rs, en, d4, d5, d6, d7); // Inicializa el objeto
   lcd con los pines definidos
4. const int LED = 9; // Define el pin para el LED
5. const int BOTON = 8; // Define el pin para el botón
6. int val; // Variable para almacenar el estado del botón
7.
8. void setup() {
9.   pinMode(LED, OUTPUT); // Configura el pin del LED como salida
```

```
10.         pinMode(BOTON, INPUT); // Configura el pin del botón como
    entrada
11.         lcd.begin(16, 2); // Configura el LCD con 16 columnas y 2
    filas
12.         lcd.print("Estado del LED:"); // Imprime el mensaje en la
    primera línea del LCD
13.         lcd.setCursor(0, 1); // Establece el cursor en la primera
    posición de la segunda línea
14.         lcd.print("APAGADO"); // Imprime "APAGADO" en la segunda
    línea
15.       }
16.
17.       void loop() {
18.         val = digitalRead(BOTON); // Lee el estado del botón
19.         if (val == HIGH) { // Si el botón está presionado
20.           lcd.clear(); // Limpia la pantalla del LCD
21.           lcd.setCursor(0, 0); // Establece el cursor en la primera
    posición de la primera línea
22.           lcd.print("Estado del LED:"); // Imprime el mensaje en la
    primera línea del LCD
23.           lcd.setCursor(0, 1); // Establece el cursor en la primera
    posición de la segunda línea
24.           if (digitalRead(LED) == LOW) { // Si el LED está apagado
25.             digitalWrite(LED, HIGH); // Enciende el LED
26.             lcd.print("ENCENDIDO"); // Imprime "ENCENDIDO" en la
    segunda línea
27.           } else { // Si el LED está encendido
28.             digitalWrite(LED, LOW); // Apaga el LED
29.             lcd.print("APAGADO"); // Imprime "APAGADO" en la
    segunda línea
30.           }
31.           delay(500); // Espera 0.5 segundos para evitar rebotes en
    el botón
32.         }
33.       }
```

iv. Trabajo independiente

Realice las siguientes tareas según lo aprendido en la sesión:

- ✓ Realice un texto deslizante que imite la siguiente secuencia y se repita:



Solución

- ✓ Dibuje una carita feliz en la matriz o cualquier otro objeto de su preferencia.

Solución

Reflexión STEAM

A nivel científico, se explora el funcionamiento básico de la pantalla de cristal líquido (LCD), lo que implica comprender los principios detrás de este dispositivo específico de visualización. La tecnología nunca deja de presenciarse pues se utilizan componentes de hardware como Arduino, potenciómetros, LED y botones, así como el desarrollo de código para controlar el LCD y realizar diversas acciones, como imprimir texto, contar segundos y mostrar el estado de un LED.

Los usuarios interiorizan más las conexiones eléctricas entre los diferentes componentes y a diseñar circuitos utilizando la protoboard, una de las implicaciones de esto es que analicen cómo seleccionar los componentes adecuados para un proyecto (idealización importante para el área ingenieril) y se rectifica con la planeación de las actividades en el trabajo independiente.

Aunque puede parecer más técnico, el diseño y la presentación de efectos visuales en la pantalla LCD, como el texto deslizante, implican un aspecto creativo que puede considerarse parte un arte digital. Acompañado de la matriz 8x8 visto en la sesión anterior, la creatividad de los usuarios puede explotarse más.

Por último, se incluyen elementos matemáticos como el contador de segundos y la impresión de una ecuación matemática paso a paso, lo que proporciona una aplicación práctica de conceptos matemáticos en un contexto tecnológico.

Sesión 5

Sensores

1:00 hora	H1-H4-H6

- [ ] Leer sensor de temperatura IDE [H1]
- [ ] Imprimir datos de sensor de temperatura en el LCD [H1]
- [ ] Leer fotorresistor IDE [H1]
- [ ] Encender LED con fotorresistor [H1]
- [ ] Leer control de luz infrarroja IDE [H1]
- [ ] Encender LED con control de luz infrarroja [H1]
- [ ] Trabajo independiente [H1 – H4 – H6]

ESPECIFICACIONES

Esta sesión se dedicará únicamente a la interpretación de datos. Para poder leer una característica se suele usar un sensor, para esta sesión utilizaremos un sensor de temperatura, un fotorresistor (sensor de intensidad de luz) y un control remoto de luz infrarroja con su respectivo receptor.

Se iniciará con el sensor de temperatura. Existen muchos tipos de sensor, algunos más precisos que otros, pero para esta sesión se utilizará el sensor LM35. Este sensor ya mide la temperatura en grados Celsius; sin embargo, al usar la función analogRead (), nos devolverá un valor entre 0 y 1023, este valor se distribuirá entre los 0V y los 5V. Por ello, para poder obtener la temperatura en grado Celsius, se debe utilizar la siguiente fórmula:

$$T = \frac{analogRead() \cdot 5 \cdot 100}{1024}$$

Ahora que ya se conoce cómo calcular la temperatura, es momento de leerla a través del IDE de Arduino.

i. Leer sensor de temperatura en el IDE

En el esquema 12 se muestra las conexiones eléctricas para la pantalla LCD. En la figura 12 se señalan los pines del sensor de temperatura LM35.

Esquema 12: leer sensor de temperatura en el IDE

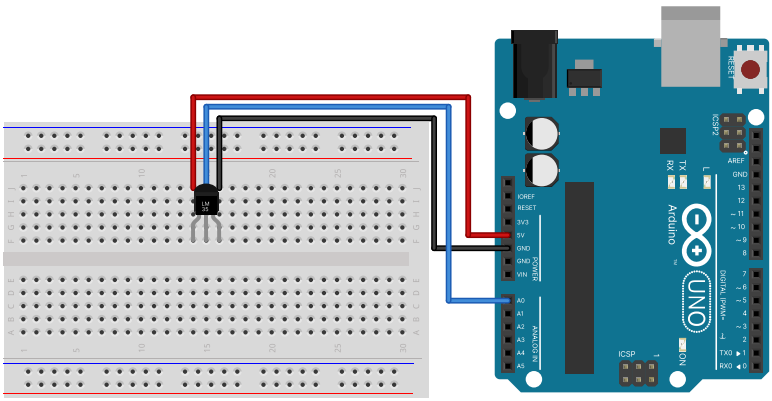
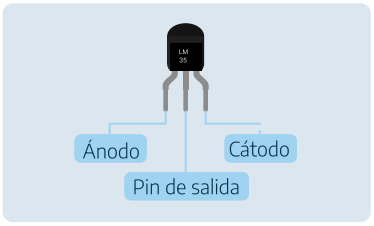


Figura 12

Pines del potenciómetro



Objetivo:

Leer datos del sensor de temperatura en el serial monitor.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers wires
4. Sensor LM35

1. Conecte el cátodo del sensor al pin tierra de Arduino.
2. Conecte el ánodo del sensor al pin de 5V.
3. Conecte el pin de salida al pin análogo A0.

Código

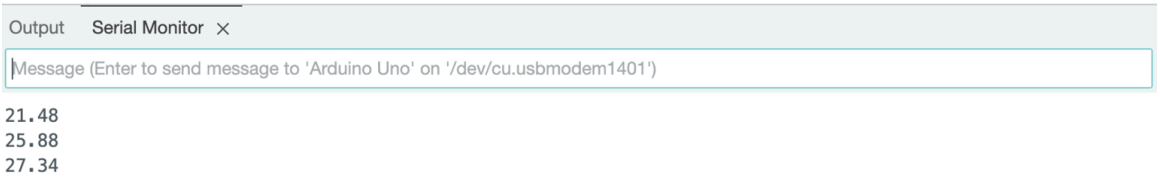
```
1. float tempC; // Variable para almacenar la temperatura en grados
   Celsius
2. int pinLM35 = A0; // Define el pin del sensor LM35
3.
4. void setup() {
5.   Serial.begin(9600); // Inicia la comunicación con el monitor
   serial
6. }
7.
8. void loop() {
9.   tempC = analogRead(pinLM35); // Lee el valor analógico del sensor
   LM35
```

```
10.         tempC = (5.0 * tempC * 100.0) / 1024.0; // Convierte el
    valor analógico a grados Celsius
11.         Serial.println(tempC); // Imprime la temperatura en grados
    Celsius en el monitor serial
12.         delay(1000); // Espera 1 segundo antes de tomar otra
    lectura
13.       }
```

Al cargar el código y visualizar el serial monitor, se pueden observar los registros de temperatura como se muestra en la figura 13.

Figura 13

Registro de temperatura en el serial monitor

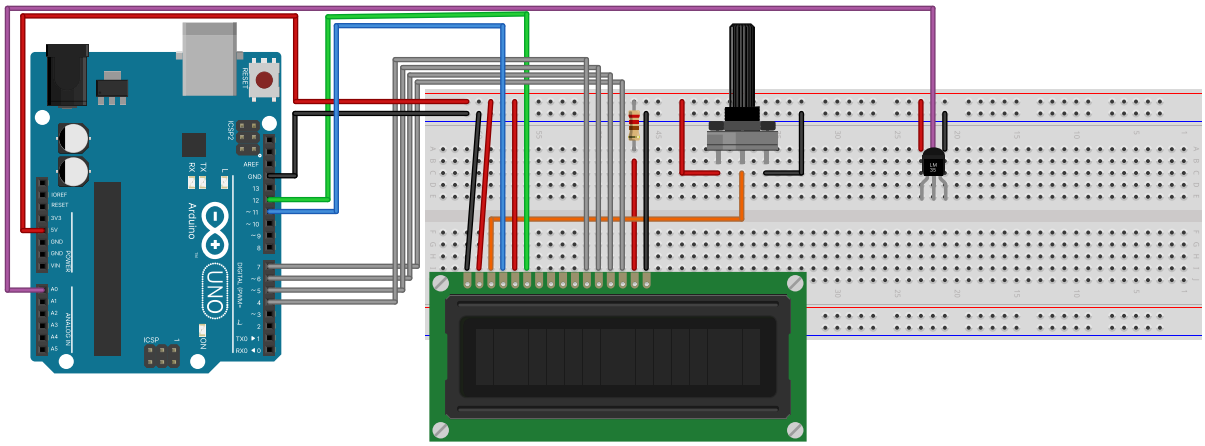


Como se puede observar, los valores fluctúan bastante. Hay formas en código que podrían permitir unos cálculos más precisos, pero la mayoría de las ocasiones depende más del componente que del código. Ahora, se procederá a imprimir esos datos que visualizamos en el serial monitor en el LCD.

ii. Imprimir datos del sensor de temperatura en el LCD

En el esquema 13 se muestra las conexiones eléctricas correspondientes a esta actividad.

Esquema 13: imprimir datos del sensor de temperatura en el LCD



Objetivo:

Mostrar en la pantalla LCD los datos que detecta el sensor de temperatura.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers wires
4. Sensor LM35
5. Resistencia 220 Ω
6. 16x2 LCD

1. Conecte un cable puente a la terminal de tierra del Arduino al canal designado para conexión a tierra y otro para el canal designado para energía.
2. Conecte el potenciómetro con sus respectivos cables puentes.
3. Conecte el sensor de temperatura con sus respectivos cables puentes y resistencia.
4. Conecte los cables puentes y resistencia necesarios para la pantalla LCD.

Código

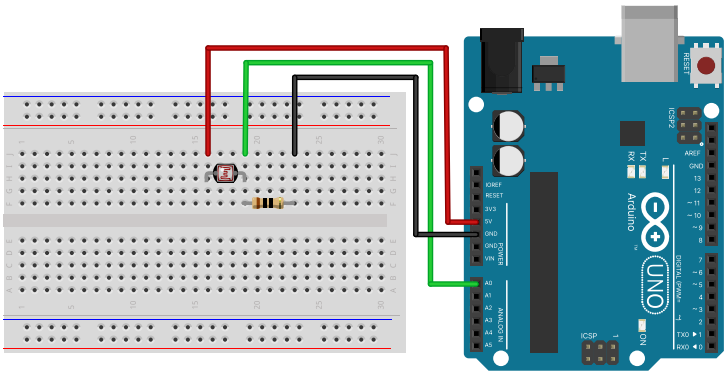
```
#include <LiquidCrystal.h>
// Define los pines para la pantalla LCD
const int rs = 11, en = 12, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7); // Crea un objeto lcd
float tempC; // Variable para almacenar la temperatura en grados Celsius
int pinLM35 = A0; // Define el pin donde se conecta el sensor LM35

void setup() {
  lcd.begin(16, 2); // Inicializa la pantalla LCD con 16 columnas y 2 filas
        lcd.print("Temperatura:"); // Muestra "Temperatura:" en la primera fila
        Serial.begin(9600); // Inicia la comunicación con el monitor serial
      }

      void loop() {
        tempC = analogRead(pinLM35); // Lee el valor del sensor LM35
        tempC = (5.0 * tempC * 100.0) / 1024.0; // Convierte el valor leído a grados Celsius
        lcd.setCursor(0, 1); // Coloca el cursor en la segunda fila
        lcd.print(tempC); // Muestra la temperatura en la pantalla LCD
        delay(1000); // Espera 1 segundo antes de leer nuevamente
      }
```

Ahora la temperatura se imprime en el LCD, ahorrando la necesidad de mirar el serial monitor. Es momento de trabajar con el fotorresistor y leer sus datos.

iii. Leer fotorresistor IDE

Para esta unidad se utilizará un CdS. Un fotorresistor CdS es un componente electrónico cuya resistencia cambia según el brillo. El sulfuro de cadmio (CdS) se utiliza como material interno y su valor de resistencia disminuye a medida que la luz que lo incide se vuelve más fuerte. El CdS no posee ánodo ni cátodo. El esquema 14 muestra las conexiones eléctricas para esta actividad.

***Esquema 14**: leer fotorresistor en el IDE*



Objetivo:

Leer datos de la fotorresistencia en el serial monitor.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers Wires
4. CdS
5. Resistencia 10k Ω

1. Conecte a un extremo del CdS un cable puente al pin de 5V del Arduino.
2. Al otro extremo del CdS, conecte un cable puente al pin A0.
3. En ese mismo extremo conecte una resistencia.
4. Al otro lado de la resistencia, conecte un cable puente al pin Tierra del Arduino.

Código

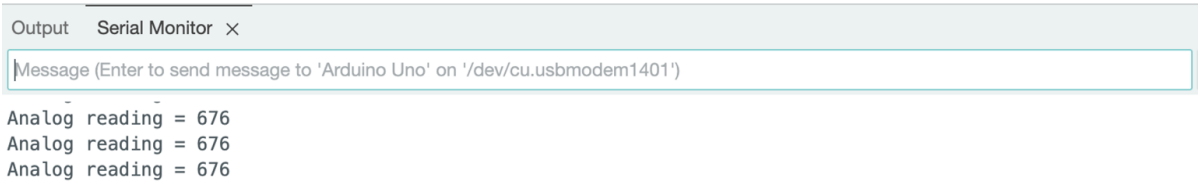
```
1. int photocellPin = A0; // Define el pin donde se conecta la
   fotorresistencia
2. int photocellReading; // Variable para almacenar la lectura
   analógica
3. void setup() {
4.   Serial.begin(9600); // Inicia la comunicación con el monitor
   serial
5. }
6. void loop() {
```

```
  photocellReading = analogRead(photocellPin); // Lee el valor de la fotorresistencia
  Serial.print("Analog reading = "); // Imprime el texto "Analog reading = " en el monitor serial
  Serial.println(photocellReading); // Imprime el valor leído en el monitor serial
        delay(1000); // Espera 1 segundo antes de leer nuevamente
    }
```

Se deben observar los datos en el serial monitor como se muestra en la figura 14.

Figura 14

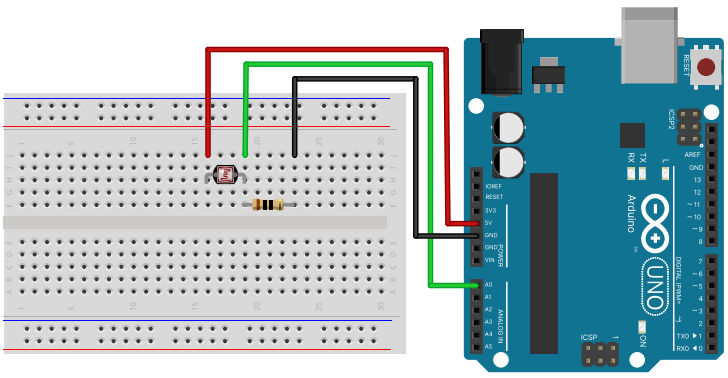
Registro de temperatura en el serial monitor



Se ha abarcado la lectura e impresión de datos, es momento de trabajar con ellos. En la siguiente actividad, se encenderá un LED según los datos obtenidos del CdS.

iv. Encender LED con fotorresistor

Esquema 15: leer fotorresistor en el IDE



1. Mantenga las conexiones de la actividad anterior.
2. Conecte la luz LED con sus respectivos cables puentes y resistencia.

Objetivo:

Leer datos de la fotorresistencia en el serial monitor.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers Wires
4. Resistencia 220 Ω
5. Resistencia 10k Ω
6. Luz LED
7. CdS

Código

```
int valorLDR = 0; // Variable para almacenar el valor leído del LDR
int pinLed1 = 8; // Pin donde está conectado el LED
int pinLDR = A0; // Pin analógico donde está conectado el LDR

void setup() {
  pinMode(pinLed1, OUTPUT); // Configura el pin del LED como salida
  Serial.begin(9600); // Inicia la comunicación con el monitor serial
}
void loop() {
        valorLDR = analogRead(pinLDR); // Lee el valor del LDR
        Serial.println(valorLDR); // Imprime el valor del LDR en el monitor serial
        delay(1000); // Espera 1 segundo antes de continuar
        if (valorLDR < 525) { // Si el valor del LDR es menor que 525
          digitalWrite(pinLed1, HIGH); // Enciende el LED
        } else if (valorLDR >= 525) { // Si el valor del LDR es mayor o igual que 525
          digitalWrite(pinLed1, LOW); // Apaga el LED
        }
      }
```

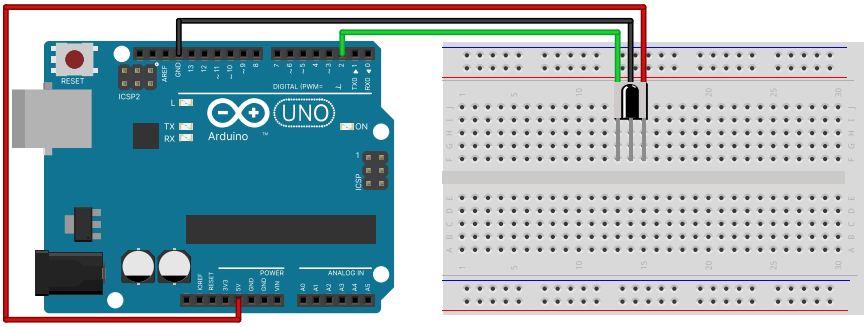
Si la luz no se enciende, puede ser porque la habitación está muy iluminada o entra mucha luz natural. En esos casos es mejor ver el serial monitor para ver el promedio de luz reconocido por el fotorresistor y variar los límites de los condicionales.

v. Leer control de luz infrarroja en el IDE

Finalmente, se trabajará con el receptor de luz infrarroja, para ello se necesita un control de luz infrarroja. Existen varios, para esta unidad didáctica se utilizará el mostrado a la derecha. Sin embargo, el receptor de IR al interpretar cada señal del control brinda un código distinto para cada botón pulsado. Lo primero que se debe hacer es reconocer cuáles códigos le corresponden a los botones.

Figura 15
Control IR

Esquema 16: leer control de luz infrarroja en el IDE



Objetivo:

Descifrar los códigos de cada botón del control

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers wires
4. Receptor IR

1. Conecte el ánodo del receptor IR al pin 5V.
2. Conecte el cátodo del receptor IR al pin tierra del Arduino.
3. Conecte al pin 2 el pin de entrada del receptor IR.

Se debe instalar la librería IRremote, para ello basta buscarlo en la opción de librerías y descargarlo. Sin embargo, para los códigos que utilizaremos en esta sesión, se debe utilizar la versión 2.0.1 (en el propio IDE puede elegir la versión de la librería).

Código

```
#include <IRremote.h>
int RECV_PIN = 2; // Pin donde está conectado el receptor IR
IRrecv irrecv(RECV_PIN); // Inicializa el receptor IR
decode_results results; // Variable para almacenar los resultados

void setup() {
  Serial.begin(9600); // Inicia la comunicación con el monitor serial
  irrecv.enableIRIn();  // Inicia el receptor IR
  irrecv.blink13(true); // Activa el parpadeo del LED en el pin 13 al recibir datos IR
      }

      void loop() {
        if (irrecv.decode(&results)) { // Si se recibe una señal IR
        Serial.println(results.value, HEX); // Imprime el valor de la señal en formato hexadecimal
        irrecv.resume(); // Prepara el receptor
        }
      }
```

Con el código anterior, se podrá observar en el serial monitor el código de cada botón al presionarlo. Si se deja presionado un botón, el serial monitor empezará a imprimir "FFFFFFFF". Una vez se identifique cada código se asociarán como se muestra en la tabla 7.

Tabla 7

Asociación de botones del control IR con su código

Botón	Código	Botón	Código
CH-	FFA25D	0	FF6897
CH	FF629D	1	FF30CF
CH+	FFE21D	2	FF18E7
|<<	FF22DD	3	FF7A85
>>|	FF02FD	4	FF10EF
>||	FFC23D	5	FF38C7
-	FFE01F	6	FF5AA5
+	FFA857	7	FF42BD
EQ	FF906F	8	FF4AB5
100+	FF9867	9	FF52AD
200+	FFB04F		

Ahora se cambia el código para que, en vez de imprimir los códigos, imprima el botón que corresponda.

Código

```
#include <IRremote.h>
int RECV_PIN = 2; // Pin donde está conectado el receptor IR
IRrecv irrecv(RECV_PIN); // Inicializa el receptor IR
decode_results results; // Variable para almacenar los resultados decodificados
unsigned long key_value = 0; // Variable para almacenar el valor de la tecla anteriormente presionada

void setup() {
  Serial.begin(9600); // Inicia la comunicación con el monitor serial
  irrecv.enableIRIn();  // Inicia el receptor IR
        irrecv.blink13(true); // Activa el parpadeo del LED en el pin 13 al recibir datos IR
      }

      void loop() {
        if (irrecv.decode(&results)) { // Si se recibe una señal IR
```

```
15.          if (results.value == 0xFFFFFFFF) { // Si el valor de la
   señal es igual a 0xFFFFFFFF (repetición)
16.          results.value = key_value; // Asigna el valor de la tecla
   anteriormente presionada
17.        }
18.        switch (results.value) { // Comprueba el valor de la señal
   recibida
19.          case 0xFFA25D:
20.            Serial.println("CH-");
21.            break;
22.          case 0xFF629D:
23.            Serial.println("CH");
24.            break;
25.          case 0xFFE21D:
26.            Serial.println("CH+");
27.            break;
28.          case 0xFF22DD:
29.            Serial.println("|<<");
30.            break;
31.          case 0xFF02FD:
32.            Serial.println(">>|");
33.            break;
34.          case 0xFFC23D:
35.            Serial.println(">||");
36.            break;
37.          case 0xFFE01F:
38.            Serial.println("-");
39.            break;
40.          case 0xFFA857:
41.            Serial.println("+");
42.            break;
43.          case 0xFF906F:
44.            Serial.println("EQ");
45.            break;
46.          case 0xFF9867:
47.            Serial.println("100+");
48.            break;
49.          case 0xFFB04F:
50.            Serial.println("200+");
51.            break;
52.          case 0xFF6897:
53.             Serial.println("0");
54.             break;
55.           case 0xFF30CF:
56.             Serial.println("1");
57.            break;
58.          case 0xFF18E7:
59.            Serial.println("2");
```

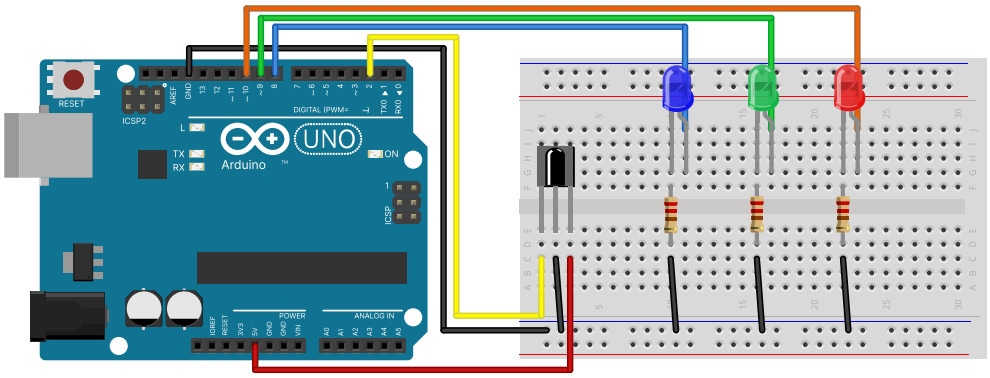
```
            break;
          case 0xFF7A85:
            Serial.println("3");
            break;
          case 0xFF10EF:
            Serial.println("4");
            break;
          case 0xFF38C7:
            Serial.println("5");
            break;
          case 0xFF5AA5:
            Serial.println("6");
            break;
           case 0xFF42BD:
             Serial.println("7");
             break;
          case 0xFF4AB5:
            Serial.println("8");
            break;
          case 0xFF52AD:
            Serial.println("9");
            break;
          }
          delay(500); // Espera 500 milisegundos
          key_value = results.value; // Actualiza el valor de la
   tecla presionada
          irrecv.resume(); // Prepara el receptor para recibir el
   siguiente valor
        }
      }
```

Se debe visualizar el serial monitor para verificar que cada botón funciona. Ahora que se puede identificar claramente cuál botón se presiona, se puede programar eventos según el botón presionado, como se muestra en la siguiente actividad.

vi. Encender LED con control de luz infrarroja

En el esquema 17 se muestran las conexiones eléctricas para esta actividad.

Esquema 17: encender LED con control de luz infrarroja



Objetivo:

Encender el LED azul al presionar el 1; el verde, al presionar el 2 y el rojo, al presionar el 3

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers Wires
4. Resistencia 220 Ω
5. Receptor IR
6. Luces LED

Código

```
#include <IRremote.h>
int RECV_PIN = 2; // Pin donde está conectado el receptor IR
IRrecv irrecv(RECV_PIN); // Inicializa el receptor IR
decode_results results; // Variable para almacenar los resultados decodificados
unsigned long key_value = 0; // Variable para almacenar el valor de la tecla anteriormente presionada
const int LED1 = 8, LED2 = 9, LED3 = 10; // Pines de los LEDs

void setup() {
  Serial.begin(9600); // Inicia la comunicación con el monitor serial
        irrecv.enableIRIn(); // Inicia el receptor IR
        irrecv.blink13(true); // Activa el parpadeo del LED en el pin 13 al recibir datos IR
        pinMode(LED1, OUTPUT); // Configura el pin LED1 como salida
        pinMode(LED2, OUTPUT); // Configura el pin LED2 como salida
        pinMode(LED3, OUTPUT); // Configura el pin LED3 como salida
      }

      void loop() {
        if (irrecv.decode(&results)) { // Si se recibe una señal IR
          if (results.value == 0xFFFFFFFF) { // Si el valor de la señal es igual a 0xFFFFFFFF (repetición)
```

```
20.          results.value = key_value; // Asigna el valor de la tecla
   anteriormente presionada
21.        }
22.        switch (results.value) { // Comprueba el valor de la señal
   recibida
23.          case 0xFF30CF:
24.            digitalWrite(LED1, HIGH);
25.            digitalWrite(LED2, LOW);
26.            digitalWrite(LED3, LOW);
27.          break;
28.          case 0xFF18E7:
29.            digitalWrite(LED1, LOW);
30.            digitalWrite(LED2, HIGH);
31.            digitalWrite(LED3, LOW);
32.          break;
33.          case 0xFF7A85:
34.            digitalWrite(LED1, LOW);
35.            digitalWrite(LED2, LOW);
36.            digitalWrite(LED3, HIGH);
37.          break;
38.          default:
39.             digitalWrite(LED1, LOW);
40.             digitalWrite(LED2, LOW);
41.             digitalWrite(LED3, LOW);
42.          }
43.          delay(500); // Espera 500 milisegundos
44.          key_value = results.value; // Actualiza el valor de la
   tecla presionada
45.          irrecv.resume(); // Prepara el receptor para recibir el
   siguiente valor
46.        }
47.      }
```

vii. Trabajo independiente

Realice las siguientes tareas según lo aprendido en la sesión:

- ✓ Imprimir los valores del fotorresistor en el LCD. Solución
- ✓ Encender LED según los datos obtenidos del sensor de temperatura. Solución
- ✓ Imprimir la secuencia de botones que se presionen en el LCD. Solución

Reflexión STEAM

En esta ocasión, se visualiza la base científica de los conceptos involucrados en la medición de la temperatura y la intensidad de la luz; a nivel tecnológico, los usuarios aprenden a utilizar dispositivos electrónicos y a programarlos para realizar diversas tareas, lo que implica comprender y utilizar tecnología de hardware y software.

Aunque de manera más sutil, se puede considerar que la presentación de los datos en una pantalla LCD y la creación de una interfaz visual para mostrar los resultados implica un aspecto artístico pues hay que diseñar primero mentalmente la forma en que se organizan los datos y se presentan en el LCD.

Aunque no se aborda explícitamente en la descripción de la sesión, el cálculo de la temperatura en grados Celsius a partir de la lectura analógica del sensor LM35 implica el uso de conceptos matemáticos, como la conversión de unidades y el uso de ecuaciones.

Sesión 6

Display 7-segmentos
Display 4x7-segmentos

1:30 horas	H1-H2-H3-H4-H5-H6

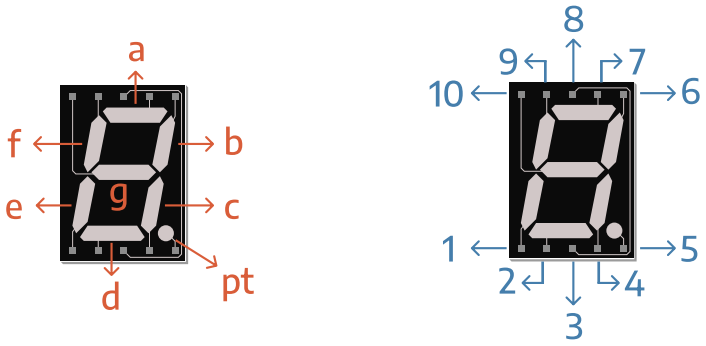
- Contador ascendente con display 7-segmentos [H1]
- Número random con display 7-segmentos [H1 – H2 – H4]
- Secuencias de números display 4x7-segmentos [H1 – H2 – H4]
- Contador hasta 9999 modulando velocidad [H1]
- Trabajo independiente [H1 – H4 – H6]

ESPECIFICACIONES

Un display 7-segmentos es un componente que tiene 7 segmentos LED, más un LED que hará de punto, indicando cómo hay que colocar el display, y siempre irá hacia abajo. Por lo tanto, se trabaja como si se tuviera 7 LED conectados al Arduino. En esta unidad, se utilizará la versión 5011AS. Al tratarse de 7 segmentos LED, se puede numerar como se muestra en la figura 16.

Figura 16

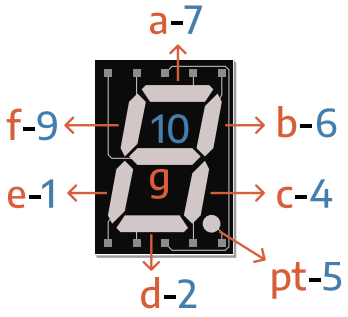
Numeración del display 7-segmentos



Además, los 10 pines con los que cuenta también se pueden numerar. Con el esquema anterior se pueden relacionar 7 pines con las 7 tiras LED como se muestra en la figura. El pin 5 corresponde al LED punto. Los pines 3 y 8 representan el ánodo o cátodo común, solo se conecta uno de ellos, según la versión de display que se maneje.

Figura 17

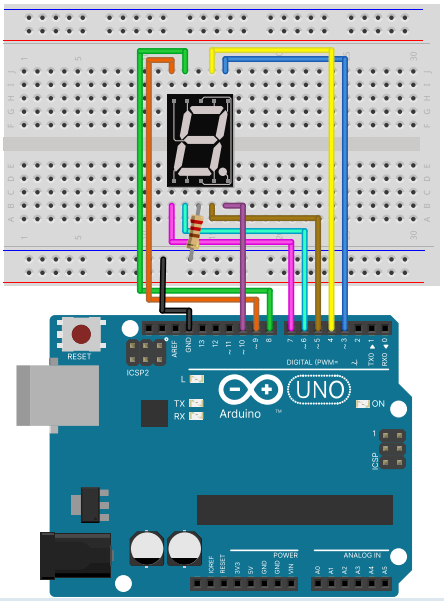
Relación 7 pines con 7 tiras



i. Contador ascendente display 7-segmentos

Es momento de conectar el display al Arduino. Como se mencionó anteriormente, cada segmento LED es independiente, por lo que cada pin de segmento LED debe tener su pin en el Arduino como se muestra en el esquema 18.

***Esquema 18**: contador ascendente display 7-segmentos*



Objetivo:

Realizar un contador del 0 al 9 en el display 7-segmentos

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers wires
4. Display 7-segmentos
5. Resistencia 220 Ω

1. Conecte del pin 3 al 10 del Arduino los pines del display siguiendo este orden 6, 7, 4, 2, 1, 9, 10, 5. De forma que el pin 6 y pin 5 del Display estén conectados al pin 3 y 10 del Arduino respectivamente.
2. Conecte una resistencia entre el pin 3 del Display y el canal tierra.
3. Conecte el pin tierra del Arduino al canal tierra de la Protoboard.

Código

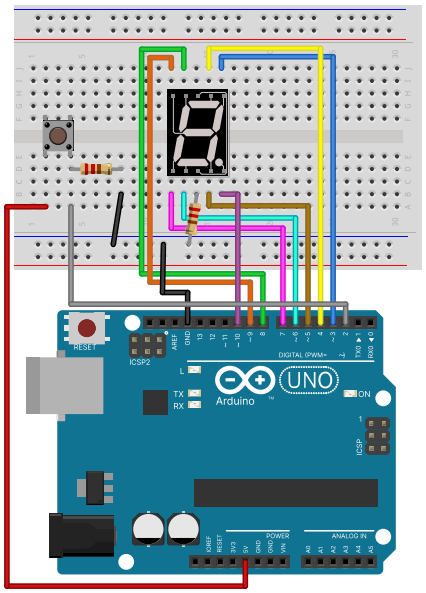
```
byte numero[10][8] = {
  { 1, 1, 1, 1, 1, 1, 0, 0 }, // 0
  { 1, 0, 1, 0, 0, 0, 0, 0 }, // 1
  { 1, 1, 0, 1, 1, 0, 1, 0 }, // 2
  { 1, 1, 1, 1, 0, 0, 1, 0 }, // 3
  { 1, 0, 1, 0, 0, 1, 1, 0 }, // 4
  { 0, 1, 1, 1, 0, 1, 1, 0 }, // 5
  { 0, 1, 1, 1, 1, 1, 1, 0 }, // 6
  { 1, 1, 1, 0, 0, 0, 0, 0 }, // 7
      { 1, 1, 1, 1, 1, 1, 1, 0 }, // 8
      { 1, 1, 1, 1, 0, 1, 1, 0 }  // 9
    };

    void setup() {
      Serial.begin(9600); // Inicia la comunicación con el monitor serial
      for (int i = 3; i < 10; i++) {
        pinMode(i, OUTPUT); // Configura los pines 3 a 9 como salidas
      }
    }

    void loop() {
      for (int num = 0; num < 10; num++) {
        for (int e = 0; e < 8; e++) {
          digitalWrite(e + 3, numero[num][e]); // Establece el estado de los pines 3 a 10 según el número actual
        }
        if (num == 9) {
          num = -1; // Reinicia el ciclo cuando llega al último número
        }
        delay(1000); // Espera 1 segundo antes de mostrar el siguiente número
      }
    }
```

Cada lista del arreglo “número” representa un número del 0 al 9, siguiendo el orden según el pin que representan: {6, 7, 4, 2, 1, 9, 10, 5}. Por lo que basta saber cuáles segmentos LED se quieren encender y poner un “1” en el pin correspondiente, de esta manera se puede hacer la forma que uno quisiera.

ii. Número random con display 7-segmentos

Para esta actividad, se mantendrá la conexión eléctrica de la actividad anterior con la diferencia que se añadirá un botón como se muestra en el esquema 19. Además, se usará una función random que se detalla más adelante.

Esquema 19: número random con display 7-segmentos



Objetivo:

Que el Display imprima un número aleatorio entre el 0 y 9 cada vez que se presione el botón.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers wires
4. Display 7-segmentos
5. Resistencia 220 Ω
6. Botón

1. Las conexiones del Display 7-Segmentos se mantienen.
2. Para el botón, conecte el ánodo al pin de 5V del Arduino.
3. Conecte una resistencia para el otro extremo del botón.
4. Conecte el pin 2 del Arduino al botón.
5. Conecte un cable puente de la resistencia al canal Tierra de la Protoboard.

Código

```
1. const int BOTON = 2;
2. int randomNumber;
3. byte numero[10][8] = {
4.   { 1, 1, 1, 1, 1, 1, 0, 0 }, // 0
5.   { 1, 0, 1, 0, 0, 0, 0, 0 }, // 1
6.   { 1, 1, 0, 1, 1, 0, 1, 0 }, // 2
7.   { 1, 1, 1, 1, 0, 0, 1, 0 }, // 3
8.   { 1, 0, 1, 0, 0, 1, 1, 0 }, // 4
9.   { 0, 1, 1, 1, 0, 1, 1, 0 }, // 5
10.       { 0, 1, 1, 1, 1, 1, 1, 0 }, // 6
11.       { 1, 1, 1, 0, 0, 0, 0, 0 }, // 7
12.       { 1, 1, 1, 1, 1, 1, 1, 0 }, // 8
```

```
    { 1, 1, 1, 1, 0, 1, 1, 0 }  // 9
  };

  void setup() {
    Serial.begin(9600); // Inicia la comunicación con el monitor serial
    for (int i = 3; i < 10; i++) {
      pinMode(i, OUTPUT); // Configura los pines 3 a 9 como salidas
    }
    pinMode(BOTON, INPUT); // Configura el pin 2 como entrada para el botón
    randomSeed(analogRead(0)); // Inicializa la generación de números aleatorios con una semilla única
  }

  void loop() {
    int valor = digitalRead(BOTON); // Lee el estado del botón
    if (valor == HIGH) { // Si el botón está presionado
      randomNumber = random(0, 10); // Genera un número aleatorio entre 0 y 9
      for (int e = 0; e < 8; e++) {
        digitalWrite(e + 3, numero[randomNumber][e]); // Establece el estado de los pines 3 a 10 según el número aleatorio
      }
      delay(500); // Espera 0.5 segundos antes de generar otro número aleatorio
    }
  }
```

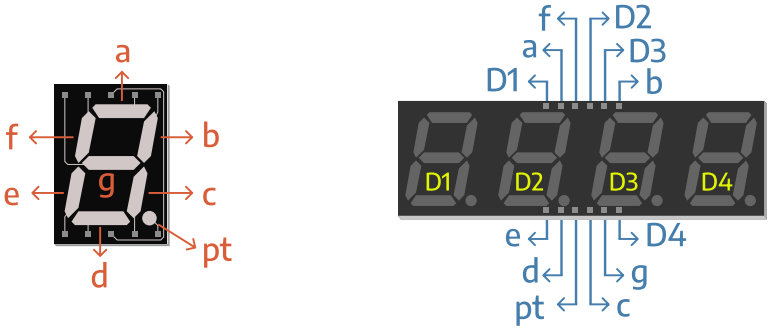
La función **random (n,m)** genera números pseudoaleatorios. El parámetro n es el número mínimo incluido del rango y el número m es máximo excluido del rango. Sin embargo, es importante agregar el código **randomSeed(analogRead(0))** en el void setup ( ). Esto permite que el "origen" del random cambie cada vez que se ejecute el código. De lo contrario, cada vez que se ejecute seguirá la misma secuencia en el mismo orden. Se reta al usuario a crear una condición que deba cumplir el número para que pueda ser impreso.

Ahora que se conoce cómo funciona el display 7-segmentos, resulta más sencillo aprender a utilizar el display 4x7-segmentos. Para ello, se requiere conocer sus partes y respectivos pines. Al igual que el display 7-segmentos, cada uno de los dígitos

se puede dividir en 7-segmentos de LED. La diferencia radica en que además de los 8 pines para las 8 tiras LED, también posee 4 pines, uno para cada dígito (D1 para el primer dígito de izquierda a derecha y D4 para el último) como se muestra en la figura 18.

Figura 18

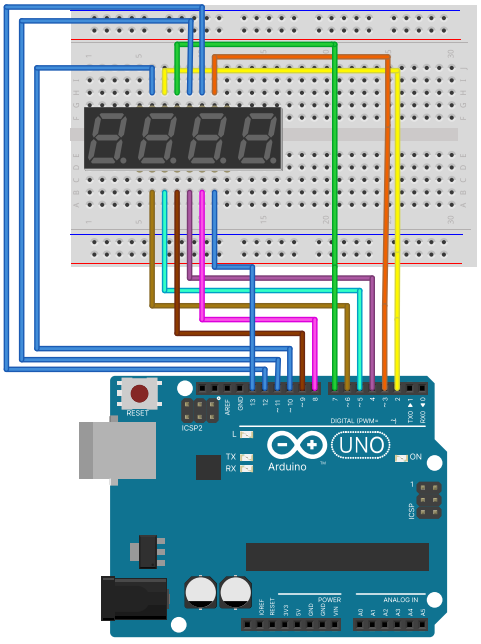
Asociación de pines en el display 4x7-segmentos



iii. Secuencia de números en el display 4x7-segmentos

Al igual que con el display 7-Segmentos, lo primero que se hará es conocer cómo imprimir en cada dígito y cada número del 0 al 9. Las coneciones eléctricas para esta actividad se muestran en el esquema 20.

***Esquema 20**: secuencia de núemros en display 4x7-segmentos*



Objetivo:

Que el display imprima un número aleatorio entre el 0 y 9 pasando por cada digito.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers wires
4. Display 4x7-segmentos

1. Conecte los pines “a” hasta el “pt” del display 4x7-segmentos a los pines del 2 al 9 del Arduino respectivamente.
2. Conecte los pines “D1” hasta el “D4” del display 4x7-segmentos a los pines del 10 al 13 del Arduino respectivamente.

Código

```
// Declaración de matriz para definir los segmentos de cada número del 0 al 9
byte numero[10][8] = {
  { 0, 0, 0, 0, 0, 0, 1, 1}, // 0
  { 1, 0, 0, 1, 1, 1, 1, 1}, // 1
  { 0, 0, 1, 0, 0, 1, 0, 1}, // 2
  { 0, 0, 0, 0, 1, 1, 0, 1}, // 3
  { 1, 0, 0, 1, 1, 0, 0, 1}, // 4
  { 0, 1, 0, 0, 1, 0, 0, 1}, // 5
  { 0, 1, 0, 0, 0, 0, 0, 1}, // 6
        { 0, 0, 0, 1, 1, 1, 1, 1}, // 7
        { 0, 0, 0, 0, 0, 0, 0, 1}, // 8
        { 0, 0, 0, 0, 1, 0, 0, 1}  // 9
      };
      // Declaración de matriz para definir qué LEDs se encienden para indicar qué dígito está siendo mostrado
      byte grupo[4][4] = {
        { 1, 0, 0, 0,},
        { 0, 1, 0, 0,},
        { 0, 0, 1, 0,},
        { 0, 0, 0, 1,}
      };
      // Inicialización de variable de control
      int cont = 0;

      void setup() {
        // Configuración de pines como salidas
        for (int i = 2; i < 14; i++) {
          pinMode(i, OUTPUT);
        }
      }

      void loop() {
        // Encendido de LEDs de grupo para indicar qué dígito se está mostrando
        for (int e = 0; e < 4; e++) {
          digitalWrite(e + 10, grupo[0][e]);
        }
```

```
36.         delay(1000); // Retardo
37.         // Incremento del contador y selección del próximo número a
    mostrar
38.         if (cont == 9){
39.           cont = 0;
40.         } else {
41.           cont = cont + 1;
42.         }
43.       // Encendido de segmentos correspondientes para mostrar el
    número actual
44.         for (int e = 0; e < 4; e++) {
45.           digitalWrite(e + 10, grupo[1][e]);
46.         }
47.         for (int e = 0; e < 8; e++) {
48.           digitalWrite(e + 2, numero[cont][e]);
49.         }
50.         delay(1000); // Retardo
51.         // Incremento del contador y selección del próximo número a
    mostrar
52.         if (cont == 9){
53.           cont = 0;
54.         } else {
55.           cont = cont + 1;
56.         }
57.         // Encendido de segmentos correspondientes para mostrar el
    número actual
58.         for (int e = 0; e < 4; e++) {
59.           digitalWrite(e + 10, grupo[2][e]);
60.         }
61.         for (int e = 0; e < 8; e++) {
62.           digitalWrite(e + 2, numero[cont][e]);
63.         }
64.         delay(1000); // Retardo
65.         // Incremento del contador y selección del próximo número a
    mostrar
66.         if (cont == 9){
67.           cont = 0;
68.         } else {
69.           cont = cont + 1;
70.         }
71.         // Encendido de segmentos correspondientes para mostrar el
    número actual
72.         for (int e = 0; e < 4; e++) {
73.           digitalWrite(e + 10, grupo[3][e]);
74.         }
75.         for (int e = 0; e < 8; e++) {
76.           digitalWrite(e + 2, numero[cont][e]);
77.         }
```

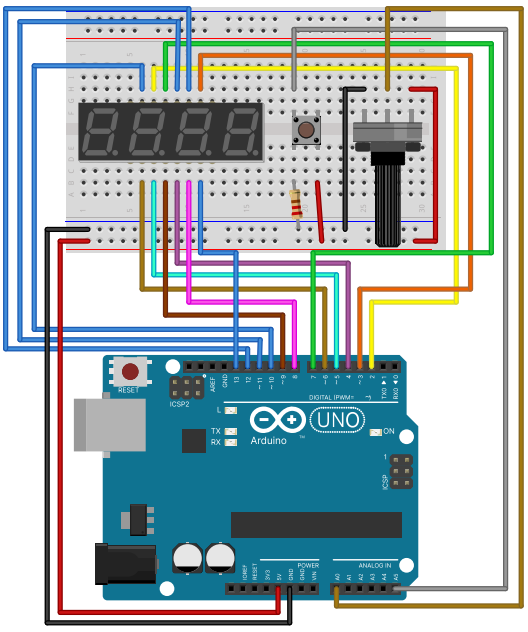
```
78.         delay(1000); // Retardo
79.         // Incremento del contador y selección del próximo número a
    mostrar
80.         if (cont == 9){
81.           cont = 0;
82.         } else {
83.           cont = cont + 1;
84.         }
85.       }
```

Al igual que con el sisplay 7-segmentos, cada lista del arreglo “numero” se refiere a un número del 0 al 9, siguiendo el orden según el pin que representan: {a, b, c, d, e, f, g, pt}. Por lo que, de igual forma, basta saber cuáles segmentos LED se quieren encender y poner un “1” en el pin correspondiente, de esta manera se puede hacer la forma que uno quisiera.

iv. Contando hasta 9999 modulando velocidad

Ahora vamos a añadir dos componentes, el botón y el potenciómetro, con la intención de conocer cómo interactúan entre ellos. Las conexiones eléctricas se muestran en el esquema 21.

Esquema 21: contando hasta 9999 modulando velocidad



Objetivo:

Crear un contador del 0 al 9999 cuya velocidad dependa del potenciómetro.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumpers wires
4. Display 4x7-segmentos
5. Botón
6. Potenciómetro
7. Resistencia 220

1. Conecte el display 4x7-segmentos como se hizo anteriormente.
2. Utilice un cable puente para conectar el pin 5V del Arduino al canal de energía de la protoboard.
3. Utilice un cable puente para conectar el pin tierra con su respectivo canal
4. Conecte el ánodo y cátodo del botón al canal de energía y tierra de la protoboard respectivamente.
5. Conecte el botón al pin A5 del Arduino.
6. Conecte el ánodo y cátodo del potenciómetro al canal de energía y tierra de la protoboard respectivamente.
7. Conecte el potenciómetro al pin A0 del Arduino en la protoboard.

Código

```
// Declaración de variables
long n = 0; // Número
int x = 100; // Divisor para obtener cada dígito del número
int del = 1000; // Retardo
int cont = 0; // Contador
// Declaración de matriz para definir los segmentos de cada número del 0 al 9 en un display de siete segmentos
byte numero[10][8] = {
  { 0, 0, 0, 0, 0, 0, 1, 1}, // 0
  { 1, 0, 0, 1, 1, 1, 1, 1}, // 1
  { 0, 0, 1, 0, 0, 1, 0, 1}, // 2
  { 0, 0, 0, 0, 1, 1, 0, 1}, // 3
  { 1, 0, 0, 1, 1, 0, 0, 1}, // 4
  { 0, 1, 0, 0, 1, 0, 0, 1}, // 5
  { 0, 1, 0, 0, 0, 0, 0, 1}, // 6
  { 0, 0, 0, 1, 1, 1, 1, 1}, // 7
  { 0, 0, 0, 0, 0, 0, 0, 1}, // 8
  { 0, 0, 0, 0, 1, 0, 0, 1}  // 9
};
// Declaración de matriz para definir qué LEDs se encienden para indicar qué dígito está siendo mostrado
byte grupo [4][4] = {
  { 1, 0, 0, 0,},
  { 0, 1, 0, 0,},
  { 0, 0, 1, 0,},
  { 0, 0, 0, 1,}
};
void setup() {
  // Configuración de pines como salidas
```

```
28.    for (int i = 2; i < 14; i++) {
29.      pinMode(i, OUTPUT);
30.    }
31.    pinMode(A5, INPUT);
32.    pinMode(A0, INPUT);
33.  }
34.
35.  void loop(){
36.    // Lectura del estado del pin A5 para reiniciar el contador
37.    int v=digitalRead(A5);
38.    if(v==1){
39.      limpiar();
40.      n=0;
41.      delayMicroseconds(5000);
42.    }
43.    // Lectura del valor analógico del pin A0
44.    int t=analogRead(A0);
45.    // Limpieza de los segmentos
46.    limpiar();
47.    // Mostrar el primer dígito del número actual
48.    digito(0);
49.    for (int e = 0; e < 8; e++) {
50.      digitalWrite(e + 2, numero[(n/x/1000)%10][e]);
51.    }
52.    delayMicroseconds(t);
53.    // Limpieza de los segmentos
54.    limpiar();
55.    // Mostrar el segundo dígito del número actual
56.    digito(1);
57.    for (int e = 0; e < 8; e++) {
58.      digitalWrite(e + 2, numero[(n/x/100)%10][e]);
59.    }
60.    delayMicroseconds(t);
61.    // Limpieza de los segmentos
62.    limpiar();
63.    // Mostrar el tercer dígito del número actual
64.    digito(2);
65.    for (int e = 0; e < 8; e++) {
66.      digitalWrite(e + 2, numero[(n/x/10)%10][e]);
67.    }
68.    delayMicroseconds(t);
69.    // Limpieza de los segmentos
70.    limpiar();
71.    // Mostrar el cuarto dígito del número actual
72.    digito(3);
73.    for (int e = 0; e < 8; e++) {
74.      digitalWrite(e + 2, numero[n/x % 10][e]);
75.    }
```

```
76.    delayMicroseconds(t);
77.    // Reiniciar el contador cuando el número llega a 9999
78.    if(n/x > 9999){
79.      n = 0;
80.      delay(5000);
81.    } else n++;
82.  }
83.  // Función para mostrar los segmentos correspondientes a un dígito
     en específico
84.  void digito(int x){
85.    for (int e = 0; e < 4; e++) {
86.      digitalWrite(e + 10, grupo[x][e]);
87.    }
88.  }
89.  // Función para apagar todos los segmentos del display
90.  void limpiar(){
91.    for (int i = 2; i < 10; i++) {
92.      digitalWrite( i, HIGH);
93.    }
94.  }
```

v. Trabajo independiente

Realice las siguientes tareas según lo aprendido en la sesión:

- ✓ Realice un contador en el display 4x7-Segmentos que vaya de par en par. Solución
- ✓ Utilice el control IR para poder imprimir un número en el display 7-segmentos. Solución
- ✓ Realice un contador descendente que inicie en 1000 y cada vez que represente un número múltiplo de 11 se debe encender un LED. Solución

Reflexión STEAM

Para esta y demás sesiones, se presencia el área científica en el uso de nuevos componentes pues se discute la estructura y partes de estos, lo que implica comprender los materiales y su comportamiento eléctrico. Particularmente para esta sesión, se evidencia al análisis el funcionamiento de los displays.

Esta sesión presenta código para controlar los displays LED. Esto implica el uso de tecnología para la programación y control de hardware. Además, con cada actividad se afianzan los principios de circuitos electrónicos.

Se diseñan sistemas para controlar los displays y se plantean desafíos como la creación de un contador ascendente y descendente, así como la generación de números aleatorios. Los usuarios se enfrentan a desafíos como la modulación de la velocidad de un contador utilizando un potenciómetro, lo que implica la resolución de problemas de ingeniería.

Aunque no se menciona directamente, el diseño de la visualización en los displays puede considerarse una forma de arte, donde se juega con la disposición de los segmentos LED para crear patrones y números legibles

A nivel matemático, se utilizan conceptos matemáticos como contar números ascendentes y descendentes, generar números aleatorios, definición de segmentos específicos para representar números.

Sesión 7

Motor y Servomotor

1:30 horas	H1- H4 -H6

ESPECIFICACIONES

- Motor paso a paso con controlador [H1]
- Motor paso a paso con controlador y batería 9V [H1]
- Servomotor [H1]
- Controlar servomotor con potenciómetro [H1]
- Trabajo independiente [H1 – H4 – H6]

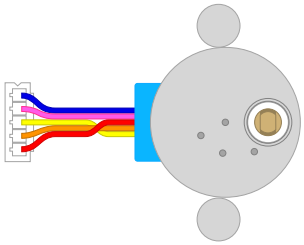
Esta sesión se concentra en el uso de motor y servomotor, la utilidad de estos componentes expande las aplicaciones de Arduino en proyectos más complejos donde la creatividad de quién manipula estos componentes puede verse explotada.

Un motor es un dispositivo que transforma la energía eléctrica en energía mecánica, empleándola para realizar una tarea que involucra movimiento. Hay diferentes tipos de motores: los de corriente continua (CC), que giran continuamente hasta que se corta la corriente; los servomotores, con un sistema que permiten moverse y detectar su posición angular siendo más precisos en un rango de movimiento de 0° a 180°; y los stepmotor o motor paso a paso, que pueden girar continuamente como motores de CC y colocarse con precisión (en pasos discretos) como servomotores.

Es necesario mencionar que los motores requieren de un “controlador” o “driver” de motor que ayuda a superar la brecha entre los voltajes proporcionados por la placa de Arduino y la necesaria para mover el motor (más alta que la proporcionada por la placa).

Para la presente unidad didáctica se utilizará la pareja de stepmotor-driver más utilizada para empezar en el mundo de motores que corresponde al modelo STEPMOTOR28BYJ-48 junto al controlador ULN2003 que se muestran en la figura 19 y la figura 20 respectivamente.

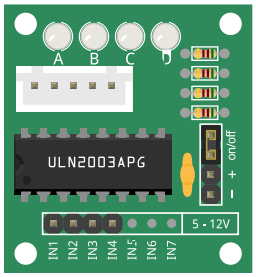
Figura 19
Motor paso a paso 28BYJ-48



El motor tiene un ángulo de paso de 5,625°, es decir, 64 pasos por vuelta usando half-step. La precisión máxima es de 4096 pasos por vuelta, equivalente a un paso de 0,088°. Si se usara el full-step se requerirían 2048 pasos para un giro completo (360°) y cada paso contribuye a un giro de 0.18°

En el driver hay cuatro diodos LED en la parte superior (A, B, C, D). El conector blanco en el medio estará conectado al motor paso a paso. Los pines IN1, IN2, IN3 e IN4 se conectan a cuatro salidas digitales del Arduino. Cuando una entrada IN sea alta, el LED correspondiente se iluminará. Los pines a la derecha son la entrada de la fuente de alimentación.

Figura 20
Driver de motor ULN2003APG

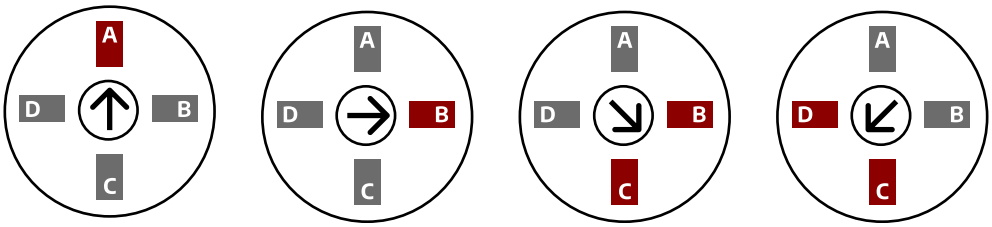


El motor internamente tiene 4 bobinas que son contraladas por los cables de color del motor (a excepción del rojo que es para energía). Cada LED del driver está asociado a una bobina.

Cuando se energiza cada bobina, se produce el movimiento del motor y se enciende el LED asociado en el driver (A-Azul, B-Rosado, C-Amarillo, D-Naranja). Se pueden energizar varias bobinas en simultáneo lo que produce diferentes pasos, por ejemplo, que sean en diagonal o en medio pasos como se muestra en la figura 21.

Figura 21

Energización de bobinas en motor paso a paso



(a) *(b)* *(c)* *(d)*

Las imágenes son ejemplo de las bobinas energizadas y cómo sería el movimiento producido, las dos imágenes (a) y (b) corresponde a un ejemplo de paso completo (full step), mientras que las (c) y (d) corresponden a un ejemplo de medio paso (half step). Así, se puede comprender la secuencia de paso completo que se muestra en la tabla 8.

Tabla 8

Secuencia de paso completo 1 y 0

Paso	A	B	C	D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

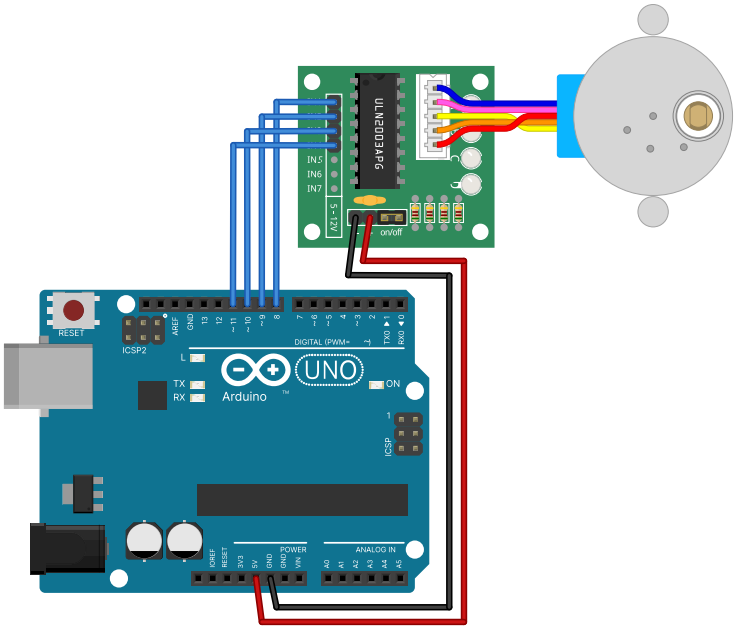
La primera bobina se energiza enviando un 1 lógico y 0 al resto, así con las demás bobinas. Se seguirá esta idea para la programación del motor.

Lo anterior refiere a lo básico sobre el motor paso a paso que se utilizará en esta unidad, un estudio más a fondo es recomendado si se desea crear proyectos más complejos que impliquen el uso del motor. Ahora, se procederá con las actividades correspondientes a la sesión.

i. Motor paso a paso con controlador

Lo primero es conectar el motor al Arduino junto con su driver como se muestra en el esquema 22.

***Esquema 22**: motor paso a paso con controlador*



Objetivo:

Girar el motor

Materiales:

1. Placa de Arduino
2. Jumper wires
3. Motor paso a paso
4. Driver

1. Conecte los pines del driver con los pines del Arduino, siguiendo la secuencia IN1-8, IN2-9, IN3-10, IN4-11 (para esta actividad).
2. Conecte jumper wires para hacer la conexión a tierra y energía del driver.
3. Conecte el motor al driver.

Existen los jumpers wire hembra (F) y machos (M), hasta el momento se ha estado utilizando el jumper macho (M-M) con picos en ambos extremos, que se usan para conectar a los diferentes pines ya sea del Arduino, protoboard u otro componente. En este caso, es necesario usar jumper hembra tanto para el motor como para el driver. Hay jumpers del tipo M-M, F-M y F-F, el starter kit contiene Jumpers del tipo F-F. Por ejemplo, se conecta un F-F al componente y un M-M de uno de los extremos del F-F al Arduino.

Con el siguiente programa se energizará una bobina por vez, se envía un nivel alto (HIGH) a la bobina que se desea energizar y nivel bajo (LOW) a las demás. En cierta forma, se está programando el motor de forma manual.

Código

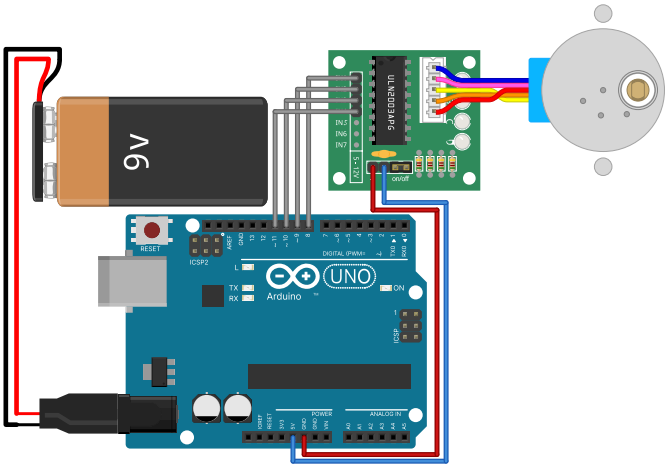
```
1. int IN1 = 8;
2. int IN2 = 9;
3. int IN3 = 10;
4. int IN4 = 11;
5.
6. int demora = 20;  // demora entre pasos, no debe ser mayor a 10 ms

7. void setup() {
8.   pinMode (IN1, OUTPUT);
9.   pinMode (IN2, OUTPUT);
10.    pinMode (IN3, OUTPUT);
11.    pinMode (IN4, OUTPUT);
12.  }

13.  void loop(){
14.    for (int i = 0; i < 512; i++) // 512*4 = 2048 pasos, es decir, se
   repiten 512 veces los 4 pasos para hacer 2048 que corresponden a una
   vuelta
15.      {
16.         digitalWrite (IN1, HIGH);  // primer paso
17.         digitalWrite (IN2, LOW);
18.         digitalWrite (IN3, LOW);
19.         digitalWrite (IN4, LOW);
20.         delay(demora);

21.         digitalWrite (IN1, LOW);  // segundo paso
22.         digitalWrite (IN2, HIGH);
23.         digitalWrite (IN3, LOW);
24.         digitalWrite (IN4, LOW);
25.         delay(demora);

26.        digitalWrite (IN1, LOW);  // tercer paso paso
27.        digitalWrite (IN2, LOW);
28.        digitalWrite (IN3, HIGH);
29.        digitalWrite (IN4, LOW);
30.        delay(demora);

31.        digitalWrite (IN1, LOW);  // cuarto paso
32.        digitalWrite (IN2, LOW);
33.        digitalWrite (IN3, LOW);
34.        digitalWrite (IN4, HIGH);
35.        delay(demora);
36.      }
37.      digitalWrite (IN1, LOW); // Se detiene por dos segundos
38.      digitalWrite (IN2, LOW);
39.      digitalWrite (IN3, LOW);
```

```
40.       digitalWrite (IN4, LOW);
41.       delay(2000);
42.   }
```

ii. Motor paso a paso con controlador y batería 9V

La batería de 9v permite energizar la placa y todo componente que requiera de energía sin necesidad de que la placa esté conectada. No existe mucho cambio respecto a cómo se han estado trabajando las actividades, la diferencia es que se debe cargar el código primero a Arduino y luego conectar la batería para que entre en funcionamiento. Las conexiones eléctricas para esta actividad se muestran en el esquema 23.

Esquema 23: motor paso a paso con driver y batería 9V



Objetivo:

Girar el motor.

Materiales:

1. Placa de Arduino
2. Jumper wires
3. Motor paso a paso
4. Driver
5. Batería 9V

1. Conecte los pines del driver con los pines del Arduino, siguiendo la secuencia IN1-8, IN2-9, IN3-10, IN4-11 (para esta actividad).
2. Conecte jumper wires para hacer la conexión a tierra y energía del driver.
3. Conecte el motor al driver.
4. Conecte la batería 9v al Arduino usando el broche correspondiente.

Con el siguiente programa energizaremos las bobinas al igual que el anterior, con la diferencia de que se manejará una matriz con 1 y 0 lógicos para controlar las bobinas. Seguirá la idea de la tabla 8. que se había estudiado con anterioridad. Con el

siguiente programa energizaremos las bobinas al igual que el anterior, con la diferencia de que se manejará una matriz con 1 y 0 lógicos para controlar las bobinas.

Código

```
int IN1 = 8;
int IN2 = 9;
int IN3 = 10;
int IN4 = 11;
int atraso = 20;
int paso [4][4]{
  {1,0,0,0},
  {0,1,0,0},
  {0,0,1,0},
    {0,0,0,1},
  };

  void setup() {
    pinMode (IN1, OUTPUT);
    pinMode (IN2, OUTPUT);
    pinMode (IN3, OUTPUT);
    pinMode (IN4, OUTPUT);
  }

  void loop(){
    for (int i = 0; i < 512; i++) // 512*4 = 2048 pasos
    {
      for(int i = 0; i < 4; i++){
        digitalWrite (IN1, paso[i][0]); // Lectura por fila matriz
        digitalWrite (IN2, paso[i][1]);
        digitalWrite (IN3, paso[i][2]);
        digitalWrite (IN4, paso[i][3]);
        delay(atraso);
      }
    }
    digitalWrite (IN1, LOW);
    digitalWrite (IN2, LOW);
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, LOW);
    delay(2000);
  }
```

Observe que se está trabajando de forma manual el motor al controlar cada bobina por separado. Si se quisiera hacer que el motor gire en dirección contraria se

deben encender las bobinas al revés. Es decir, si están encendidas las bobinas A-B-C-D, encenderlas en la secuencia D-C-B-A.

Por otro lado, sin entrar en trabajo directo con las bobinas, se puede programar el motor usando librerías, en el siguiente código se muestra un ejemplo.

Código

```
#include <Stepper.h>  // Incluir la librería
int pasosporvuelta = 2048; // Se requieren 2048 pasos por vuelta
Stepper motor(pasosporvuelta,8,10,9,11); // se especifican los pasos
y los pines donde de conexión del motor con el Arduino
void setup() {
  motor.setSpeed (10); // Definir velocidad (tiempo entre pasos)
}
void loop() {
  motor.step(pasosporvuelta);
  delay(2000);
    motor.step(-pasosporvuelta); // Una vuelta en dirección contraria
    delay(2000);
  }
```

El código anterior permite dar una vuelta en dirección de las manecillas del reloj, luego de 2 segundos da una vuelta contraria a las manecillas del reloj, y así sucesivamente. Ahora, para controlar un poco más la posición del motor, con el siguiente código se puede ingresar la cantidad de grados que se desea que le motor gire, se habilita una ventana que permite esto. Además, se muestra una conversión de cuántos pasos corresponden al grado de giro. Configura el serial monitor en “No Line Ending” en vez de “New Line”, de lo contrario imprimirá de una línea extra.

Código

```
#include <Stepper.h>
int pasosporvuelta = 2048;
Stepper motor(pasosporvuelta,8,10,9,11);
void setup() {
  Serial.begin(9600);
  motor.setSpeed (10);
}
void loop() {
  if(Serial.available()){
      int grados = Serial.parseInt();
      int pasos = map(grados, 0, 360, 0, pasosporvuelta); // Mapear los grados y pasos de movimiento del motor
      motor.step(pasos);
      Serial.println(String(grados)+"corresponden a "+String(pasos)+"pasos");
  // Para imprimir un texto que indique la conversión
   }
  }
```

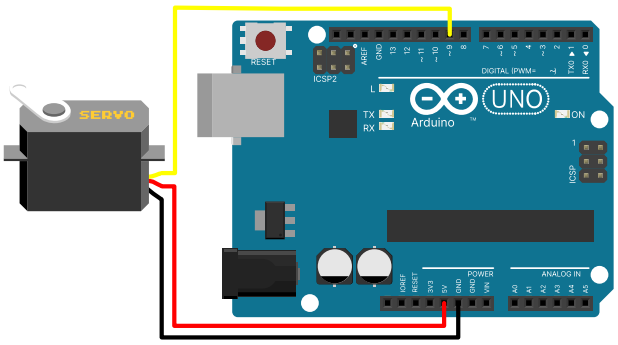
El último componente que se abarcará en la unidad didáctica es el Servomotor, particularmente el modelo SG90. Un servomotor, al igual que el motor, en un componente que convierte la energía para completar tareas que requieran movimiento. Sin embargo, el servomotor tiene un rango de movimiento limitado (0° a 180°) que compensa con la posibilidad de controlar la velocidad y la precisión de movimiento.

iii. Servomotor

Lo primero es instalar la librería "**Servo.h**", esta permite controlar de manera más sencilla el servomotor, las conexiones eléctricas se muestran en el esquema 24. Para esta primera actividad, se conocerá cómo manejar el ángulo de giro. El servomotor SG90 posee tres cables de distintos colores:

- Rojo =Alimentación
- Café = Tierra
- Naranja= Señal PWM (Modulación de ancho de pulso)

Esquema 24: servomotor



Objetivo:

Girar el motor

Materiales:

1. Placa de Arduino
2. Jumper wires
3. Servomotor

1. Conecte el cable de energía del servomotor al pin 5V del Arduino.
2. Conecte el cable tierra del servomotor al pin Tierra del Arduino.
3. Conecte la señal PWM del servomotor al pin 9 del Arduino.

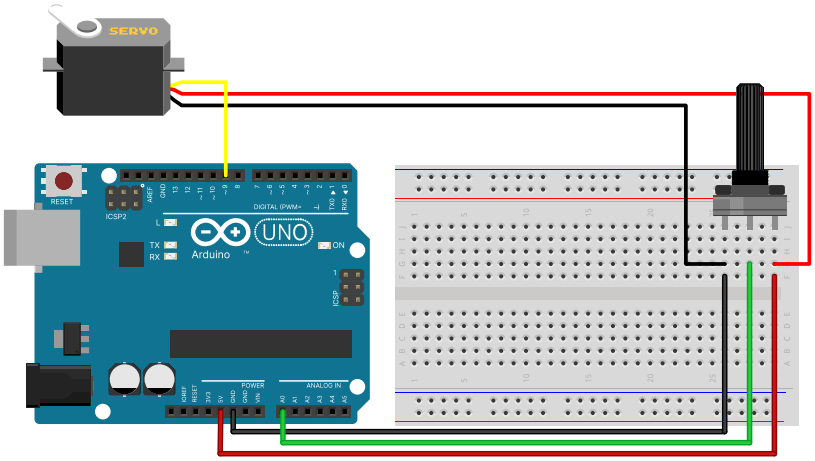
Código

```
// Incluye la biblioteca Servo.h para controlar servomotores
#include <Servo.h>
// Crea un objeto de tipo Servo llamado myservo
Servo myservo;
// Define el ángulo inicial del servomotor
int angulo = 90

void setup() {
  // Asocia el pin 9 al objeto myservo
  myservo.attach(9);
}

void loop() {
  // Establece el ángulo del servomotor
  myservo.write(angulo);
  // Espera 15 milisegundos
  delay(15);
}
```

Para variar el ángulo de giro, se debe cambiar la variable ángulo y volver a cargar el código. Para evitar ese proceso, ahora se va a determinar al valor del giro con el potenciómetro.

iv. Controlar servomotor con potenciómetro

Esquema 25: controlar servomotor con potenciómetro



Objetivo:

Determinar el ángulo del servomotor con potenciómetro.

Materiales:

1. Placa de Arduino
2. Jumper wires
3. Servomotor

Código

```
// Incluye la biblioteca Servo.h para controlar servomotores
#include <Servo.h>
// Crea un objeto de tipo Servo llamado myservo
Servo myservo;
// Define el pin al que está conectado el potenciómetro
int potpin = 0;
// Variable para almacenar el valor leído del potenciómetro
int val;
void setup() {
  // Asocia el pin 9 al objeto myservo
  myservo.attach(9);
}
void loop() {
  // Lee el valor del potenciómetro
  val = analogRead(potpin);
  // Mapea el valor leído al rango de 0 a 180 grados
  val = map(val, 0, 1023, 0, 180);
  // Establece el ángulo del servomotor según el valor del potenciómetro
  myservo.write(val);
  delay(15);
}
```

v. Trabajo independiente

Realice las siguientes tareas según lo aprendido en la sesión:

- ✓ Controlar velocidad de motor con potenciómetro. Solución
- ✓ Controlar que el motor se detenga al presionar un botón. Solución
- ✓ Utilice el control IR para determinar el ángulo de movimiento de servomotor. Solución

Reflexión STEAM

Se aborda la ciencia detrás del funcionamiento de los motores estudiados. Se explican conceptos como la conversión de energía eléctrica en energía mecánica, la secuencia de pasos en un motor paso a paso, y la precisión y rango de movimiento de un servomotor.

Esta sesión puede despertar un aspecto artístico en la creatividad necesaria para diseñar y construir proyectos que involucren el uso de motores. La manera en que se combinan los componentes mecánicos y electrónicos puede tener un elemento estético. Se puede evidenciar, por ejemplo, con una de las tareas que se desarrollará más adelante, la creación de una ruleta aleatoria, donde se pueden añadir elementos al motor para dar la impresión de una ruleta de circo.

Además, se utilizan conceptos matemáticos como el ángulo de paso en el motor paso a paso, la precisión en pasos por vuelta, y la conversión de grados a pasos para controlar el movimiento de los motores. Se emplea la función *map()* en Arduino para mapear valores analógicos a un rango específico, lo que implica una comprensión básica de funciones y relaciones matemáticas.

Sesión 8

Arduino y matemáticas

1:30 horas	H1-H2-H3-H4-H5-H6

ESPECIFICACIONES

- Dado digital con matriz 8x8 [H1 – H2]
- Calculadora de sumas y restas [H1 – H2]
- Ruleta aleatoria [H1 – H2]
- Trabajo independiente [H1 – H2 – H3 – H4 – H5 – H6]

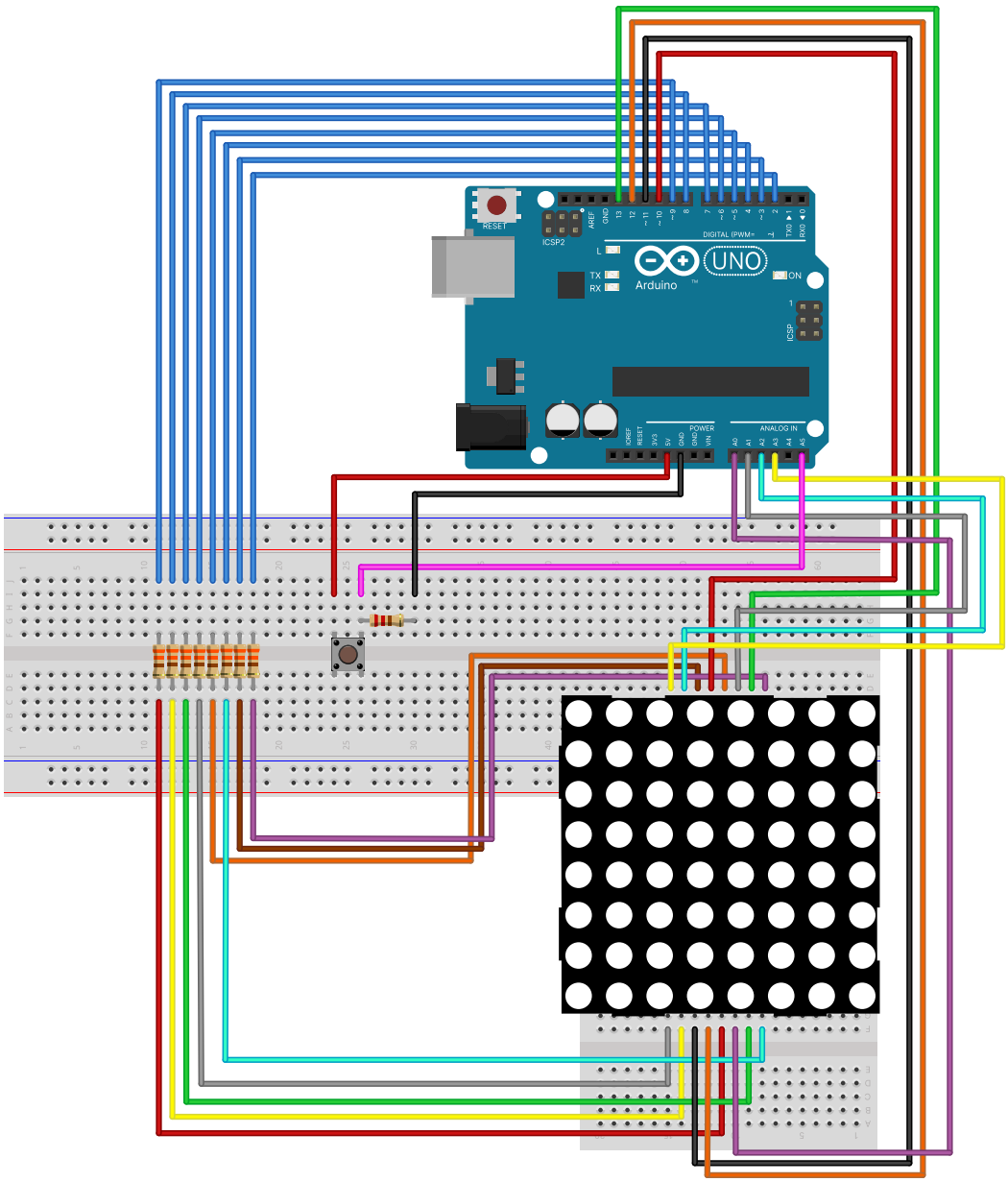
¡Perfecto, se ha logrado! La sesión 8 marca el comienzo de la última etapa de esta unidad didáctica. Como se discutió previamente, el enfoque de esta unidad ha sido en el desarrollo de habilidades STEAM. Ahora, como futuros docentes de matemáticas, es el momento de proponer actividades para quiénes serán sus estudiantes.

En esta sesión, se presentarán tres actividades que podrían implementarse en el aula, para ello se utilizarán varios componentes que se han usado durante la unidad. Sin embargo, el verdadero valor radica en la capacidad de cada docente para desplegar su creatividad y diseñar sus propias propuestas.

i. Dado digital con matriz 8x8

Para esta primera actividad se realizará un “dado digital” con la matriz 8x8 para que muestre la representación de un número al presionar un botón. Las conexiones para proceder con el dado digital se muestran en esquema 26. Esta y las demás tareas restantes se consideran conocimientos obtenidos de sesiones anteriores, sin embargo, se puede verificar el funcionamiento de cada componente realizando un repaso en su respectiva sesión.

***Esquema 26**: dado digital matriz 8x8*



Objetivo:

La matriz 8x8 debe simular una cara de un dado de seis caras cada vez que se presiona el botón.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumper wires
4. Resistencia 220 Ω
5. Resistencia 330 Ω
6. Botón
7. Matriz 8x8

1. Conecte un cable puente a la terminal de tierra del Arduino al canal designado para conexión a tierra y otro para el canal designado para energía.
2. Conecte la matriz 8x8 con sus respectivos cables puentes y resistencia.
3. Conecte los cables puentes y resistencia necesarios para el botón.

Código

```
#define R1 2
#define R2 3
#define R3 4
#define R4 5
#define R5 6
#define R6 7
#define R7 8
#define R8 9

#define C1 10
  #define C2 11
  #define C3 12
  #define C4 13
  #define C5 A0
  #define C6 A1
  #define C7 A2
  #define C8 A3

  const int BOTON=A5;
  int random1;

  int UNO  [8][8] ={
    {1,1,1,1,1,1,1,1},
    {1,1,1,1,1,1,1,1},
    {1,1,1,1,1,1,1,1},
    {1,1,1,0,0,1,1,1},
    {1,1,1,0,0,1,1,1},
    {1,1,1,1,1,1,1,1},
    {1,1,1,1,1,1,1,1},
    {1,1,1,1,1,1,1,1},
  };

  int DOS  [8][8] ={
    {1,1,1,1,1,1,1,1},
    {1,1,1,1,1,1,1,1},
    {1,1,1,1,1,0,0,1},
    {1,1,1,1,1,0,0,1},
    {1,0,0,1,1,1,1,1},
```

```
35.     {1,0,0,1,1,1,1,1},
36.     {1,1,1,1,1,1,1,1},
37.     {1,1,1,1,1,1,1,1},
38.   };
39.   int TRES  [8][8] ={
40.     {1,1,1,1,1,1,0,0},
41.     {1,1,1,1,1,1,0,0},
42.     {1,1,1,1,1,1,1,1},
43.     {1,1,1,0,0,1,1,1},
44.     {1,1,1,0,0,1,1,1},
45.     {1,1,1,1,1,1,1,1},
46.     {0,0,1,1,1,1,1,1},
47.     {0,0,1,1,1,1,1,1},
48.   };

49.   int CUATRO  [8][8] ={
50.     {1,1,1,1,1,1,1,1},
51.     {1,0,0,1,1,0,0,1},
52.     {1,0,0,1,1,0,0,1},
53.     {1,1,1,1,1,1,1,1},
54.     {1,1,1,1,1,1,1,1},
55.     {1,0,0,1,1,0,0,1},
56.     {1,0,0,1,1,0,0,1},
57.     {1,1,1,1,1,1,1,1},
58.   };

59.   int CINCO  [8][8] ={
60.     {0,0,1,1,1,1,0,0},
61.     {0,0,1,1,1,1,0,0},
62.     {1,1,1,1,1,1,1,1},
63.     {1,1,1,0,0,1,1,1},
64.     {1,1,1,0,0,1,1,1},
65.     {1,1,1,1,1,1,1,1},
66.     {0,0,1,1,1,1,0,0},
67.     {0,0,1,1,1,1,0,0},
68.   };

69.   int SEIS  [8][8] ={
70.     {1,0,0,1,1,0,0,1},
71.     {1,0,0,1,1,0,0,1},
72.     {1,1,1,1,1,1,1,1},
73.     {1,0,0,1,1,0,0,1},
74.     {1,0,0,1,1,0,0,1},
75.     {1,1,1,1,1,1,1,1},
76.     {1,0,0,1,1,0,0,1},
77.     {1,0,0,1,1,0,0,1},
78.   };
79.   void setup() {
```

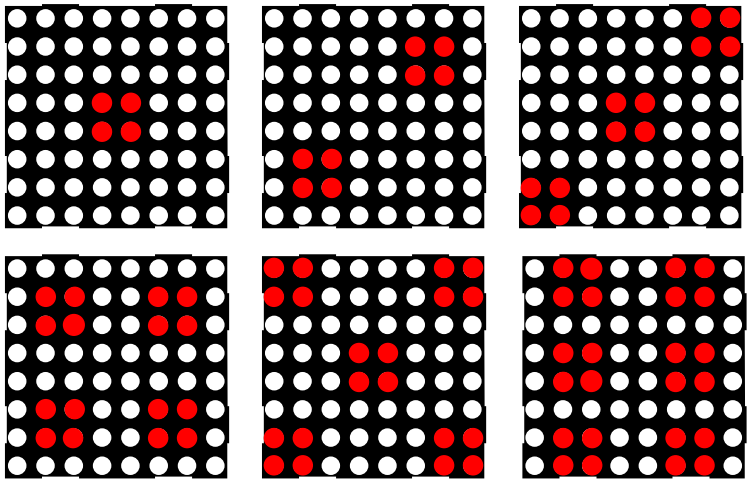
```
80.     pinMode(BOTON,INPUT);
81.     pinMode(R1, OUTPUT);
82.     pinMode(R2, OUTPUT);
83.     pinMode(R3, OUTPUT);
84.     pinMode(R4, OUTPUT);
85.     pinMode(R5, OUTPUT);
86.     pinMode(R6, OUTPUT);
87.     pinMode(R7, OUTPUT);
88.     pinMode(R8, OUTPUT);
89.     pinMode(C1, OUTPUT);
90.     pinMode(C2, OUTPUT);
91.     pinMode(C3, OUTPUT);
92.     pinMode(C4, OUTPUT);
93.     pinMode(C5, OUTPUT);
94.     pinMode(C6, OUTPUT);
95.     pinMode(C7, OUTPUT);
96.     pinMode(C8, OUTPUT);
97.     randomSeed(analogRead(0));
98.   }

99.   void loop() {
100.  if (digitalRead(BOTON)==HIGH){
101.    random1 = (random(1,7));
102.  }
103.  for (int fila = 0; fila < 8; fila++) {
104.    digitalWrite(R1+fila, HIGH);
105.      if (random1 == 1) {
106.        digitalWrite(C1, UNO[fila][0]);
107.        digitalWrite(C2, UNO[fila][1]);
108.        digitalWrite(C3, UNO[fila][2]);
109.        digitalWrite(C4, UNO[fila][3]);
110.        digitalWrite(C5, UNO[fila][4]);
111.        digitalWrite(C6, UNO[fila][5]);
112.        digitalWrite(C7, UNO[fila][6]);
113.        digitalWrite(C8, UNO[fila][7]);
114.      } else if (random1 == 2) {
115.       digitalWrite(C1, DOS[fila][0]);
116.       digitalWrite(C2, DOS[fila][1]);
117.       digitalWrite(C3, DOS[fila][2]);
118.       digitalWrite(C4, DOS[fila][3]);
119.       digitalWrite(C5, DOS[fila][4]);
120.       digitalWrite(C6, DOS[fila][5]);
121.       digitalWrite(C7, DOS[fila][6]);
122.       digitalWrite(C8, DOS[fila][7]);
123.      }
124.      else if (random1 == 3) {
125.        digitalWrite(C1, TRES[fila][0]);
126.        digitalWrite(C2, TRES[fila][1]);
```

```
127.       digitalWrite(C3, TRES[fila][2]);
128.       digitalWrite(C4, TRES[fila][3]);
129.       digitalWrite(C5, TRES[fila][4]);
130.       digitalWrite(C6, TRES[fila][5]);
131.       digitalWrite(C7, TRES[fila][6]);
132.       digitalWrite(C8, TRES[fila][7]);
133.     } else if (random1 == 4) {
134.       digitalWrite(C1, CUATRO[fila][0]);
135.       digitalWrite(C2, CUATRO[fila][1]);
136.       digitalWrite(C3, CUATRO[fila][2]);
137.       digitalWrite(C4, CUATRO[fila][3]);
138.       digitalWrite(C5, CUATRO[fila][4]);
139.       digitalWrite(C6, CUATRO[fila][5]);
140.       digitalWrite(C7, CUATRO[fila][6]);
141.       digitalWrite(C8, CUATRO[fila][7]);
142.     } else if (random1 == 5) {
143.       digitalWrite(C1, CINCO[fila][0]);
144.       digitalWrite(C2, CINCO[fila][1]);
145.       digitalWrite(C3, CINCO[fila][2]);
146.       digitalWrite(C4, CINCO[fila][3]);
147.       digitalWrite(C5, CINCO[fila][4]);
148.       digitalWrite(C6, CINCO[fila][5]);
149.       digitalWrite(C7, CINCO[fila][6]);
150.       digitalWrite(C8, CINCO[fila][7]);
151.     } else if (random1 == 6) {
152.       digitalWrite(C1, SEIS[fila][0]);
153.       digitalWrite(C2, SEIS[fila][1]);
154.       digitalWrite(C3, SEIS[fila][2]);
155.       digitalWrite(C4, SEIS[fila][3]);
156.       digitalWrite(C5, SEIS[fila][4]);
157.       digitalWrite(C6, SEIS[fila][5]);
158.       digitalWrite(C7, SEIS[fila][6]);
159.       digitalWrite(C8, SEIS[fila][7]);
160.     }
161.     delay(2);
162.     digitalWrite(R1, LOW);
163.     digitalWrite(R2, LOW);
164.     digitalWrite(R3, LOW);
165.     digitalWrite(R4, LOW);
166.     digitalWrite(R5, LOW);
167.     digitalWrite(R6, LOW);
168.     digitalWrite(R7, LOW);
169.     digitalWrite(R8, LOW);
170.   }
171. }
```

Al ejecutar el código y presionar el botón, se debe apreciar una de las caras que se muestran en la figura 22.

Figura 22

Caras del dado digital

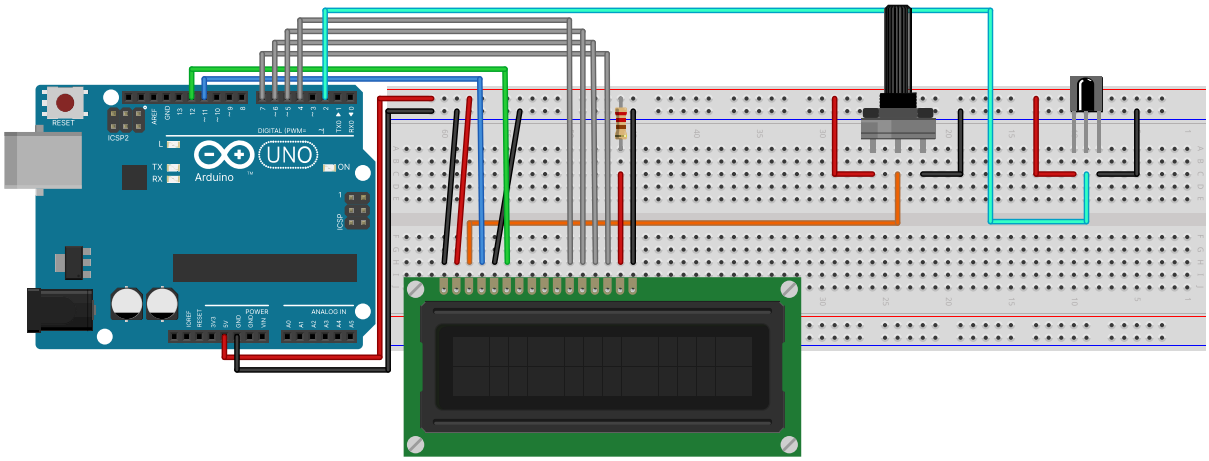


Los dados siempre serán una gran oportunidad para abarcar contenidos de probabilidad. Particularmente, la probabilidad en todos sus enfoques: intuitivo, subjetivo, frecuencial o teórico. Para variar la actividad, se pueden realizar dados con más caras para afianzar mejor el aprendizaje. Los aspectos como contenidos específicos o momentos de uso en el aula quedan a conveniencia y/o disposición de quién se disponga a implementar la actividad.

ii. Calculadora de sumas y restas

En esta actividad se utilizará el display LCD y el receptor infrarrojo con su respectivo control, las conexiones eléctricas correspondientes se muestran en el esquema 27. En sesiones previas, se ha logrado que sucedan ciertos eventos al presionar los botones del control, es momento de subir de nivel.

***Esquema 27**: calculadora de sumas y restas*



Objetivo:

Crear una calculadora de sumas y restas utilizando el control para evidenciar la igualdad en el display.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumper wires
4. Resistencia 220 Ω
5. Display LCD
6. Potenciómetro
7. Receptor IR

El control infrarrojo servirá como teclado para la calculadora. Los números se utilizan como se muestran en el control, para mostrar el resultado de la operación se presiona el botón EQ y para eliminar algún dígito insertado se presiona la tecla PREV.

1. Conecte un cable puente a la terminal de tierra del Arduino al canal designado para conexión a tierra y otro para el canal designado para energía.
2. Conecte el Display LCD con sus respectivos cables puentes y resistencia.
3. Conecte los cables puentes y resistencia necesarios para el receptor IR.

Código

```
#include <IRremote.h>
#include <LiquidCrystal.h>

int RECV_PIN = 2;
IRrecv irrecv(RECV_PIN);
decode_results  results;
unsigned long key_value=0;
const int rs = 11, en = 12, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

int result = 0;
int op = 0;

void setup(){
Serial.begin(9600);
irrecv.enableIRIn();  // Start the receiver
irrecv.blink13(true);
lcd.begin(16, 2);
}

void loop() {
if (irrecv.decode(&results))  {
Serial.println(results.value, HEX);
irrecv.resume(); // Receive  the next value
if (results.value == 0xFFFFFFFF){
results.value = key_value;}
switch (results.value){
case 0xFF6897:
if (op == 0) {
result = 0;
} else if (op == 1) {
result = result + 0;
} else if (op == 2) {
result = result - 0;
}
lcd.print("0");
break;
case 0xFF30CF:
if (op == 0) {
result = 1;
} else if (op == 1) {
result = result + 1;
} else if (op == 2) {
result = result - 1;
}
lcd.print("1");
```

```
47.   break;
48.   case 0xFF18E7:
49.   if (op == 0) {
50.   result = 2;
51.   } else if (op == 1) {
52.   result = result + 2;
53.   } else if (op == 2) {
54.   result = result - 2;
55.   }
56.   lcd.print("2");
57.   break;
58.   case 0xFF7A85:
59.   if (op == 0) {
60.   result = 3;
61.   } else if (op == 1) {
62.   result = result + 3;
63.   } else if (op == 2) {
64.   result = result - 3;
65.   }
66.   lcd.print("3");
67.   break;
68.   case 0xFF10EF:
69.   if (op == 0) {
70.   result = 4;
71.   } else if (op == 1) {
72.   result = result + 4;
73.   } else if (op == 2) {
74.   result = result - 4;
75.   }
76.   lcd.print("4");
77.   break;
78.   case 0xFF38C7:
79.   if (op == 0) {
80.   result = 5;
81.   } else if (op == 1) {
82.   result = result + 5;
83.   } else if (op == 2) {
84.   result = result - 5;
85.   }
86.   lcd.print("5");
87.   break;
88.   case 0xFF5AA5:
89.   if (op == 0) {
90.   result = 6;
91.   } else if (op == 1) {
92.   result = result + 6;
93.   } else if (op == 2) {
94.   result = result - 6;
```

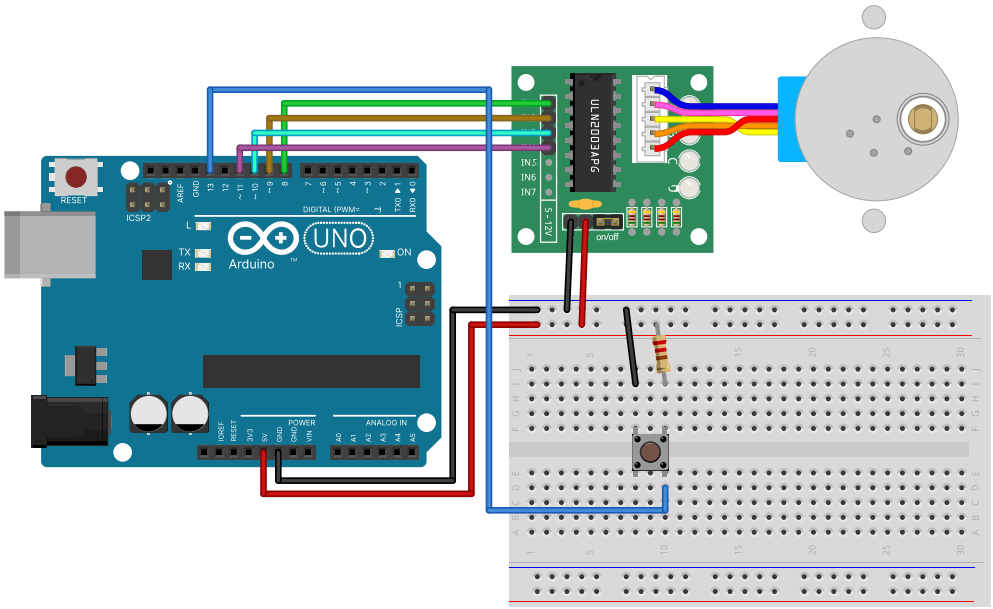
```
  }
  lcd.print("6");
  break;
  case 0xFF42BD:
  if (op == 0) {
result = 7;
} else if (op == 1) {
result = result + 7;
} else if (op == 2) {
result = result - 7;
}
lcd.print("7");
break;
case 0xFF4AB5:
if (op == 0) {
result = 8;
} else if (op == 1) {
result = result + 8;
} else if (op == 2) {
result = result - 8;
}
lcd.print("8");
break;
case 0xFF52AD:
if (op == 0) {
result = 9;
} else if (op == 1) {
result = result + 9;
} else if (op == 2) {
result = result - 9;
}
lcd.print("9");
break;
case 0xFFA857:
op = 1;
lcd.print("+");
break;
case 0xFFE01F:
if (op == 2) {
op = 1;
} else {
op = 2;
}
lcd.print("-");
break;
case 0xFF906F:
lcd.print("=");
lcd.print(result);
```

```
143.  break;
144.  case 0xFF22DD:
145.  lcd.clear();
146.  op=0;
147.  result=0;
148.  break;
149.  }
150.  delay(500);
151.  key_value = results.value;
152.  irrecv.resume();
153.  }
154.  }
```

iii. Ruleta aleatoria

En esta actividad se utilizará un botón y el motor junto a su driver, las conexiones eléctricas correspondientes se muestran en el esquema 28.

***Esquema 28**: ruleta aleatoria*



Objetivo:

Lograr que el motor se detenga en un momento aleatorio, una vez presionado el botón.

Materiales:

1. Placa de Arduino
2. Protoboard
3. Jumper wires
4. Resistencia 220 Ω
5. Botón
6. Motor 28BJY-48
7. Driver

Código

```
#include <Stepper.h>

int pasosporvuelta = 2048;
int boton = 13;
Stepper motor(pasosporvuelta,8,10,9,11);

void setup() {
  motor.setSpeed (10);
  randomSeed(analogRead(0));
  }
  void loop() {
    int presionado = digitalRead(boton);
    if (presionado == LOW){
      int grados = random(360,1000);
      int pasos = map(grados, 0, 360, 0, pasosporvuelta); // Mapear los grados y pasos de movimiento del motor
    motor.step(pasos);
   }
}
```

iv. Trabajo Independiente

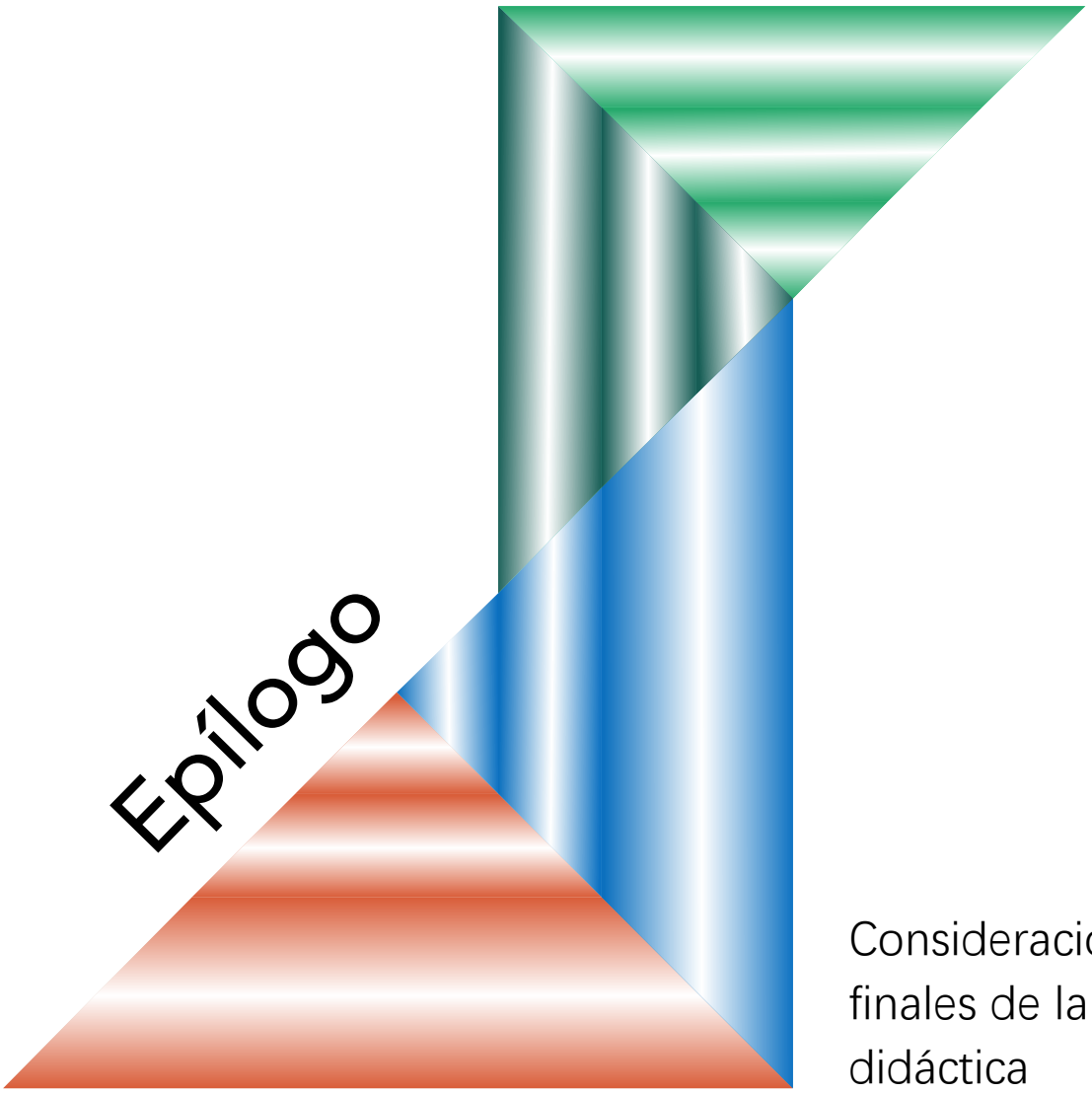
El trabajo independiente para esta última sesión se recomienda realizar en grupos y consiste en la creación de una actividad que sea de apoyo en la imparticiónde algún contenido de matemática como por ejemplo el tema de probabilidad con los dados virtuales. Queda abierto a los usuarios de la unidad.

Reflexión STEAM

En esta sesión, se puede observar una integración de conocimientos obtenidos en las sesiones anteriores. La presencia de las áreas STEAM se demuestra de forma más integral, como un todo. Se abordan conceptos de probabilidad y aleatoriedad en la simulación de lanzamientos de dados y en la generación de resultados aleatorios. Se utilizan plataformas de hardware como Arduino y componentes electrónicos para construir los dispositivos, como el dado digital y la calculadora de sumas y restas.

Se programa el Arduino para controlar el comportamiento de los dispositivos, lo que implica el uso de tecnología y lenguajes de programación. Se diseñan y montan circuitos electrónicos en la protoboard, lo que requiere habilidades para realizar conexiones eléctricas. Se emplea la resolución de problemas y el diseño de sistemas para crear dispositivos funcionales y resolver desafíos técnicos. La creatividad juega un papel importante en el diseño y la personalización de las actividades.

Las actividades están intrínsecamente relacionadas con conceptos matemáticos como probabilidad, operaciones aritméticas y mapeo de datos. Se fomenta el pensamiento lógico y la resolución de problemas matemáticos a través de la programación y la manipulación de datos.



Epílogo

Consideraciones finales de la unidad didáctica

Esta sección presenta algunos aspectos generales y reflexiones de los autores que se consideran importantes sobre la creación y lo que significaría una ejecución de la unidad, en esta parte se externalizan algunas ideas para amarrar o interiorizar el sentido de interdisciplinariedad que esta unidad representa y objetivos académicos personales de los autores.

Aspectos generales

La concepción del proceso de integración de disciplinas se vuelve evidente incluso antes de la ejecución de la unidad didáctica. Antes de que los usuarios comiencen a trabajar con este material, se espera que hayan internalizado conocimientos provenientes de diversas áreas, como didáctica y pedagogía, tecnologías y matemáticas. Estos conocimientos previos no solo son requisitos, sino que también se convierten en los cimientos sobre los cuales se construye la comprensión y el éxito en la aplicación de la unidad.

En este sentido, el proceso de integración de disciplinas se presenta como una necesidad para abordar los desafíos de un mundo cada vez más interconectado. Los usuarios no solo deben ser competentes en su área principal de estudio, sino que también deben tener la capacidad de articular y aplicar conocimientos de disciplinas aparentemente dispares. Esto no solo enriquece su comprensión del tema en cuestión, sino que también fomenta una mentalidad interdisciplinaria que es crucial en un entorno profesional y académico en constante evolución.

Al mismo tiempo, es importante ser consciente de que, con la presente unidad, los usuarios finales se adentran en un nuevo mundo, el de la electrónica. Si bien los conocimientos básicos son fundamentales para los alcances del proyecto, se espera que los usuarios adopten una mentalidad de "dar un paso extra". Esto implica no conformarse con lo establecido, sino buscar constantemente nuevas formas de innovar y expandir los límites de lo que se considera posible. Para aquellos usuarios que han

incursionado en el mundo de la electrónica, la unidad representa una oportunidad desafiante para profundizar aún más en su comprensión y dominio de este campo, a la vez que pueden aprovechar la estructura y los recursos proporcionados para expandir su repertorio de habilidades y descubrir nuevas áreas de interés y especialización.

Así, el proceso de integración de disciplinas no solo se manifiesta en la implementación práctica de la unidad, sino que también se arraiga en la mentalidad y preparación previa de los usuarios. Es un recordatorio de la importancia de abrazar la diversidad de conocimientos y perspectivas, así como de mantener una actitud de exploración y búsqueda de nuevos horizontes en el camino hacia el progreso y la excelencia.

Reflexiones

El surgimiento de esta unidad didáctica nace de una experiencia propia de los autores en una educación STEAM empírica con el apoyo de docentes del Instituto Tecnológico de Costa Rica. Esa experiencia propició la discusión de aspectos generales como los usuarios finales, objetivos de aprendizaje, actividades que los futuros docentes podrían incorporar en sus aulas como parte de distintas estrategias didácticas, entre otros. Lo anterior, desde un punto de vista empático pensando en compañeros que han tenido una educación similar a la de los creadores.

Cada paso dado en la creación de este material ha sido guiado por una profunda convicción en el poder transformador de la educación STEAM. Los autores creen en la interdisciplinariedad como un pilar de cambio en la educación. Al conectar las ciencias, la tecnología, la ingeniería, las artes y las matemáticas, no solo fortalecen el entendimiento de conceptos abstractos, sino que también cultivan la curiosidad y la capacidad de resolver problemas de manera holística.

Además, esta Unidad Didáctica no se limita a despertar un interés en las matemáticas, sino que busca cultivar una pasión genuina por el conocimiento. A través

de actividades prácticas y desafiantes se sumerge a los usuarios en el proceso de descubrimiento y exploración, adquiriendo con sus propias manos el conocimiento que les permita ser futuros agentes de cambio innovadores en los procesos educativos.

Finalmente, los autores confían plenamente en el potencial motivador de este material tanto para docentes como para estudiantes. Al proporcionar herramientas y recursos para la implementación de prácticas pedagógicas innovadoras, se aspira a despertar una nueva generación de educadores apasionados y comprometidos con el desarrollo de habilidades STEAM en las aulas.

Referencias

Aguirre, J. P. S., Vaca, V. D. C. C., y Vaca, M. C. (2019). *Educación Steam: entrada a la sociedad del conocimiento*. Ciencia Digital, 3(3.4.), 212-227.

Ale Herrera. (s.f.). *Arduino proyectos avanzados* [Lista de reproducción]. Youtube. https://youtube.com/playlist?list=PLVR0tjl2fOQKCRyLKdlbKBVD4fUBgUINM&si=0Fr15-Hv0JCQAuIa

Álvaro González. (16 de abril del 2016). *Mano robótica: proyecto Arduino (grupo 9 URJC)* [Video]. Youtube. https://www.youtube.com/watch?v=iueZPRbO0ig

Consejo Nacional de Rectores. (2023). *Noveno Informe Estado de la Educación. Programa Estado de la Nación.* https://estadonacion.or.cr/?informes=informe-estado-de-la-educacion-2023

Departamento de Orientación Educativa y Vocacional. (2022). *Estrategia Nacional de Educación STEAM. Ministerio de Educación Pública.* https://www.mep.go.cr/educatico/estrategia-nacional-educacion-steam#:~:text=Es%20una%20estrategia%20que%20promueve,ingl%C3%A9s)%20en%20sus%20proyectos%20vocacionales

Fer Informática. (12 de enero de 2022). *Sistema de riego con Arduino UNO | paso a paso | proyecto escolar* [Video]. Youtube. https://www.youtube.com/watch?v=IF9qlw6TVJI&t=3s

Mendiola, M. (2018). *Revisión de escenarios digitales de aprendizaje. En: Construcción social de una cultura digital educativas.* Universidad Pedagógica Nacional/Universidad Autónoma Metropolitana/SOMESE. 5-18.

Robot UNO. (8 de octubre del 2020). *Grúa con Ardiuno || proyecto con servomotores || para principiantes [explicado paso a paso]* [Video]. Youtube. https://www.youtube.com/watch?v=B2IwaLmHDEI&t=3s

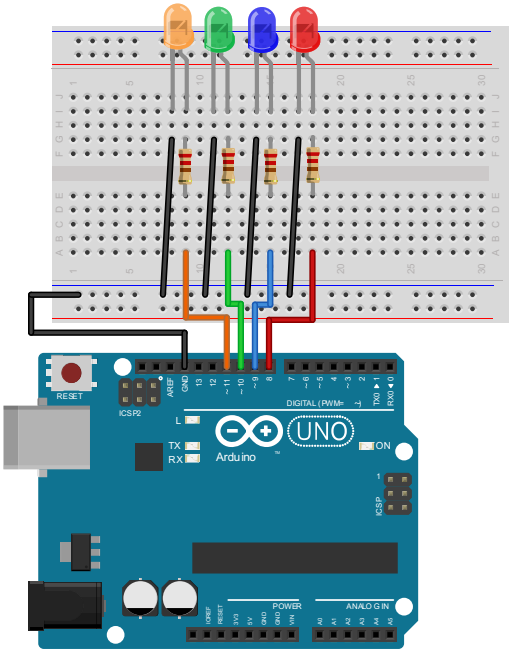
Román, M., García, C. y Carrera, F. (2023). *Trayectorias educativas de mujeres graduadas en STEM. Investigación realizada para el Noveno Informe del Estado de la Educación. CONARE, PEN.*

Segarra-Vera, X. A., Vera-Champang, S. G., y Vera-Vélez, M. L. (2024). *Potenciando el Aprendizaje con Estrategias Didácticas Innovadoras: Un Enfoque STEAM*. MQRInvestigar, *8*(1), 4913-4931. http://www.investigarmqr.com/ojs/index.php/mqr/article/view/1151

Soluciones del trabajo independiente

Sesión 1. *Actividad 1*

Esquema



Código

```
int pinLed1 = 8;
int pinLed2 = 9;
int pinLed3 = 10;
int pinLed4 = 11;
int tiempo = 100;

void setup() {
   pinMode(pinLed1, OUTPUT);
   pinMode(pinLed2, OUTPUT);
   pinMode(pinLed3, OUTPUT);
   pinMode(pinLed4, OUTPUT);
}

void loop() {
   digitalWrite(pinLed1, HIGH);
   delay(tiempo);
   digitalWrite(pinLed1, LOW);
   delay(tiempo);
   digitalWrite(pinLed2, HIGH);
   delay(tiempo);
   digitalWrite(pinLed2, LOW);
   delay(tiempo);
   digitalWrite(pinLed3, HIGH);
```

```
24.   delay(tiempo);
25.   digitalWrite(pinLed3, LOW);
26.   delay(tiempo);
27.   digitalWrite(pinLed4, HIGH);
28.   delay(tiempo);
29.   digitalWrite(pinLed4, LOW);
30.   delay(tiempo);
31.   digitalWrite(pinLed3, HIGH);
32.   delay(tiempo);
33.   digitalWrite(pinLed3, LOW);
34.   delay(tiempo);
35.   digitalWrite(pinLed2, HIGH);
36.   delay(tiempo);
37.   digitalWrite(pinLed2, LOW);
38.   delay(tiempo);
39.   digitalWrite(pinLed1, HIGH);
40.   delay(tiempo);
41.   digitalWrite(pinLed1, LOW);
42.   delay(tiempo);
43. }
```

Sesión 1. *Actividad 2*

Esquema

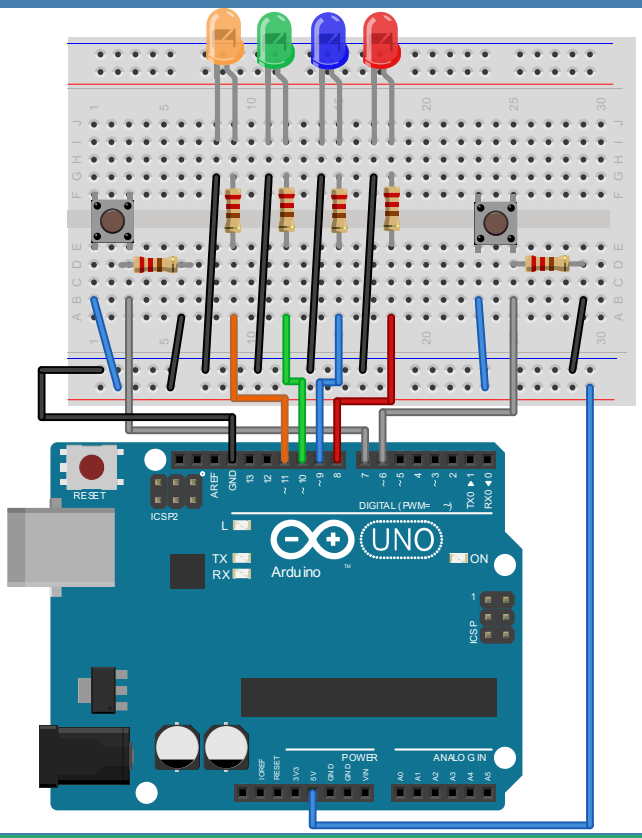
Mismo de Actividad 1

Código

```
1.  int pinLed1 = 8;
2.  int pinLed2 = 9;
3.  int pinLed3 = 10;
4.  int pinLed4 = 11;
5.  int tiempo = 400;
6.
7.  void setup() {
8.     pinMode(pinLed1, OUTPUT);
9.     pinMode(pinLed2, OUTPUT);
10.    pinMode(pinLed3, OUTPUT);
11.    pinMode(pinLed4, OUTPUT);
12. }
13. void loop() {
14.    digitalWrite(pinLed2, HIGH);
15.    digitalWrite(pinLed4, HIGH);
16.    delay(tiempo);
17.    digitalWrite(pinLed2, LOW);
18.    digitalWrite(pinLed4, LOW);
19.    digitalWrite(pinLed1, HIGH);
20.    digitalWrite(pinLed3, HIGH);
21.    delay(tiempo);
22.    digitalWrite(pinLed3, LOW);
23.    digitalWrite(pinLed1, LOW);
24.    digitalWrite(pinLed1, HIGH);
25.    digitalWrite(pinLed2, HIGH);
26.    delay(tiempo);
27.    digitalWrite(pinLed3, HIGH);
28.    digitalWrite(pinLed1, LOW);
29.    digitalWrite(pinLed4, HIGH);
30.    digitalWrite(pinLed2, LOW);
31.    delay(tiempo);
32.    digitalWrite(pinLed3, LOW);
33.    digitalWrite(pinLed1, LOW);
34.    digitalWrite(pinLed4, LOW);
35.    digitalWrite(pinLed2, LOW);
36.    }
```

Sesión 2. *Actividad 1*

Esquema



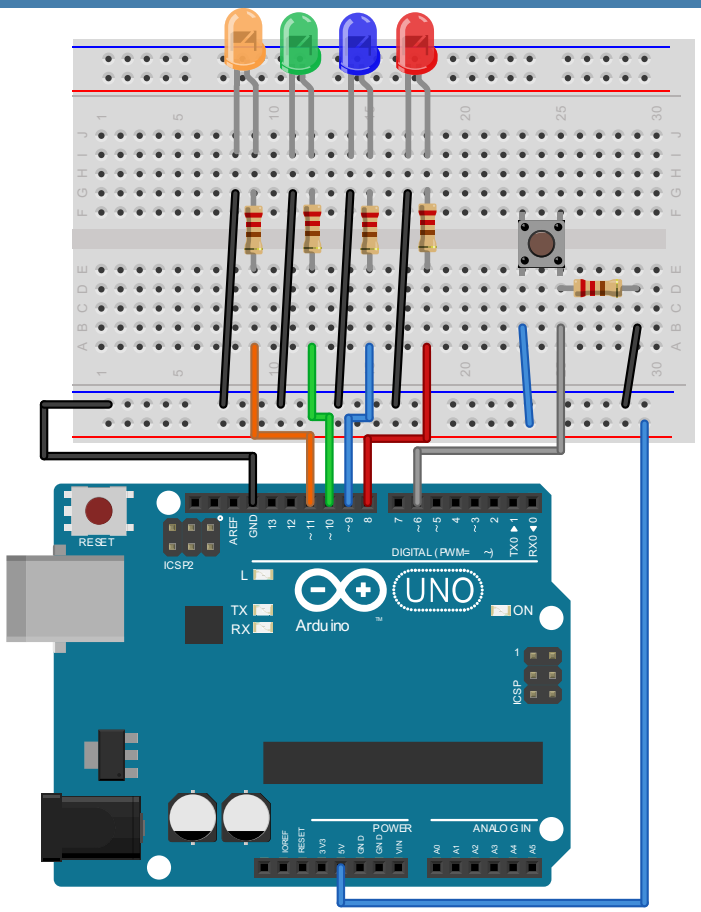
Código

```
1.  int pinLed1 = 8;
2.  int pinLed2 = 9;
3.  int pinLed3 = 10;
4.  int pinLed4 = 11;
5.  int tiempo = 100;
6.  const int BOTON_I = 7;
7.  const int BOTON_D = 6;
8.
9.  void setup() {
10.    pinMode(pinLed1, OUTPUT);
11.    pinMode(pinLed2, OUTPUT);
12.    pinMode(pinLed3, OUTPUT);
13.    pinMode(pinLed4, OUTPUT);
14.    pinMode(BOTON_I, INPUT);
15.    pinMode(BOTON_D, INPUT);
16. }
17.
18. void loop() {
19.    int val_I = digitalRead(BOTON_I);
20.    int val_D = digitalRead(BOTON_D);
21.    if (val_D == HIGH) {
22.       digitalWrite(pinLed1, HIGH);
23.       delay(tiempo);
```

```
24.      digitalWrite(pinLed1, LOW);
25.      delay(tiempo);
26.      digitalWrite(pinLed2, HIGH);
27.      delay(tiempo);
28.      digitalWrite(pinLed2, LOW);
29.      delay(tiempo);
30.      digitalWrite(pinLed3, HIGH);
31.      delay(tiempo);
32.      digitalWrite(pinLed3, LOW);
33.      delay(tiempo);
34.      digitalWrite(pinLed4, HIGH);
35.      delay(tiempo);
36.      digitalWrite(pinLed4, LOW);
37.   }
38.   if (val_I == HIGH) {
39.      digitalWrite(pinLed4, HIGH);
40.      delay(tiempo);
41.      digitalWrite(pinLed4, LOW);
42.      delay(tiempo);
43.      digitalWrite(pinLed3, HIGH);
44.      delay(tiempo);
45.      digitalWrite(pinLed3, LOW);
46.      delay(tiempo);
47.      digitalWrite(pinLed2, HIGH);
48.      delay(tiempo);
49.      digitalWrite(pinLed2, LOW);
50.      delay(tiempo);
51.      digitalWrite(pinLed1, HIGH);
52.      delay(tiempo);
53.      digitalWrite(pinLed1, LOW);
54.   }
55.}
```

Sesión 2. *Actividad 2*

Esquema



Código

```
int pinLed1 = 8;
int pinLed2 = 9;
int pinLed3 = 10;
int pinLed4 = 11;
int tiempo = 200;
const int BOTON_D = 6;
int SECUENCIA = 0;

void setup() {
   pinMode(pinLed1, OUTPUT);
   pinMode(pinLed2, OUTPUT);
   pinMode(pinLed3, OUTPUT);
   pinMode(pinLed4, OUTPUT);
   pinMode(BOTON_D, INPUT);
}

void loop() {
   int val_D = digitalRead(BOTON_D);
   if (val_D == HIGH) {
      SECUENCIA +=1;
      if ( SECUENCIA == 4) {
```

```
22.         SECUENCIA = 0;
23.       }
24.    }
25.    if (SECUENCIA==1){
26.       digitalWrite(pinLed3, LOW);
27.       digitalWrite(pinLed1, LOW);
28.       digitalWrite(pinLed4, LOW);
29.       digitalWrite(pinLed2, LOW);
30.       digitalWrite(pinLed2, HIGH);
31.       digitalWrite(pinLed4, HIGH);
32.    }
33.    if (SECUENCIA==2){
34.       digitalWrite(pinLed3, LOW);
35.       digitalWrite(pinLed1, LOW);
36.       digitalWrite(pinLed4, LOW);
37.       digitalWrite(pinLed2, LOW);
38.       digitalWrite(pinLed1, HIGH);
39.       digitalWrite(pinLed3, HIGH);
40.    }
41.    if (SECUENCIA==3){
42.       digitalWrite(pinLed3, LOW);
43.       digitalWrite(pinLed1, LOW);
44.       digitalWrite(pinLed4, LOW);
45.       digitalWrite(pinLed2, LOW);
46.       digitalWrite(pinLed1, HIGH);
47.       digitalWrite(pinLed2, HIGH);
48.    }
49.    delay(100);
50. }
```

Sesión 3. *Actividad 1*

Esquema

Mismo de esquema 8: conexión eléctrica de la matriz 8x8.

Código

```
#define R1 2
#define R2 3
#define R3 4
#define R4 5
#define R5 6
#define R6 7
#define R7 8
#define R8 9
#define C1 10
#define C2 11
#define C3 12
#define C4 13
#define C5 A0
#define C6 A1
#define C7 A2
#define C8 A3

void setup() {
   pinMode(R1, OUTPUT);
   pinMode(R2, OUTPUT);
   pinMode(R3, OUTPUT);
   pinMode(R4, OUTPUT);
   pinMode(R5, OUTPUT);
   pinMode(R6, OUTPUT);
   pinMode(R7, OUTPUT);
   pinMode(R8, OUTPUT);
   pinMode(C1, OUTPUT);
   pinMode(C2, OUTPUT);
   pinMode(C3, OUTPUT);
   pinMode(C4, OUTPUT);
   pinMode(C5, OUTPUT);
   pinMode(C6, OUTPUT);
   pinMode(C7, OUTPUT);
   pinMode(C8, OUTPUT);
   digitalWrite(C1, HIGH);
   digitalWrite(C2, HIGH);
   digitalWrite(C3, HIGH);
   digitalWrite(C4, HIGH);
   digitalWrite(C5, HIGH);
   digitalWrite(C6, HIGH);
   digitalWrite(C7, HIGH);
   digitalWrite(C8, HIGH);
}

```

```
45. void loop() {
46.   for (int fila = 0; fila < 8; fila++) {
47.     digitalWrite(R8 - fila, HIGH);
48.     for (int columna = 0; columna < 8; columna++) {
49.       digitalWrite(C8 - columna, LOW);
50.       delay(100);
51.       digitalWrite(C8 - columna, HIGH);
52.     }
53.     delay(2);
54.     digitalWrite(R1, LOW);
55.     digitalWrite(R2, LOW);
56.     digitalWrite(R3, LOW);
57.     digitalWrite(R4, LOW);
58.     digitalWrite(R5, LOW);
59.     digitalWrite(R6, LOW);
60.     digitalWrite(R7, LOW);
61.     digitalWrite(R8, LOW);
62.   }
63. }
```

Sesión 3. *Actividad 2*

Esquema

Mismo de esquema 8: conexión eléctrica de la matriz 8x8.

Código

```
#define R1 2
#define R2 3
#define R3 4
#define R4 5
#define R5 6
#define R6 7
#define R7 8
#define R8 9
#define C1 10
#define C2 11
#define C3 12
#define C4 13
#define C5 A0
#define C6 A1
#define C7 A2
#define C8 A3

int Matriz[8][8] ={
   {1, 0, 0, 0, 0, 0, 0, 1},
   {0, 1, 1, 1, 1, 1, 1, 0},
   {0, 1, 0, 1, 1, 0, 1, 0},
   {0, 1, 1, 1, 1, 1, 1, 0},
   {0, 1, 0, 1, 1, 0, 1, 0},
   {0, 1, 1, 0, 0, 1, 1, 0},
   {0, 1, 1, 1, 1, 1, 1, 0},
   {1, 0, 0, 0, 0, 0, 0, 1},
};

void setup() {
   pinMode(R1, OUTPUT);
   pinMode(R2, OUTPUT);
   pinMode(R3, OUTPUT);
   pinMode(R4, OUTPUT);
   pinMode(R5, OUTPUT);
   pinMode(R6, OUTPUT);
   pinMode(R7, OUTPUT);
   pinMode(R8, OUTPUT);
   pinMode(C1, OUTPUT);
   pinMode(C2, OUTPUT);
   pinMode(C3, OUTPUT);
   pinMode(C4, OUTPUT);
   pinMode(C5, OUTPUT);
   pinMode(C6, OUTPUT);
   pinMode(C7, OUTPUT);
```

```
  pinMode(C8, OUTPUT);
}


void loop() {
  for (int fila = 0; fila < 8; fila++) {
    digitalWrite(R1 + fila, HIGH);
    digitalWrite(C1, Matriz[fila][0]);
    digitalWrite(C2, Matriz[fila][1]);
    digitalWrite(C3, Matriz[fila][2]);
    digitalWrite(C4, Matriz[fila][3]);
    digitalWrite(C5, Matriz[fila][4]);
    digitalWrite(C6, Matriz[fila][5]);
    digitalWrite(C7, Matriz[fila][6]);
    digitalWrite(C8, Matriz[fila][7]);
    delay(2);
    digitalWrite(R1, LOW);
    digitalWrite(R2, LOW);
    digitalWrite(R3, LOW);
    digitalWrite(R4, LOW);
    digitalWrite(R5, LOW);
    digitalWrite(R6, LOW);
    digitalWrite(R7, LOW);
    digitalWrite(R8, LOW);
  }
}
```

Sesión 3. *Actividad 3*

Esquema

Mismo de esquema 8: conexión eléctrica de la matriz 8x8.

Código

```
#define R1 2
#define R2 3
#define R3 4
#define R4 5
#define R5 6
#define R6 7
#define R7 8
#define R8 9
#define C1 10
#define C2 11
#define C3 12
#define C4 13
#define C5 A0
#define C6 A1
#define C7 A2
#define C8 A3

int position = 0;
int PRUEBA[8][31] = {
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
{1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1},
{1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1},
{1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1},
{1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1},
{1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1},
{1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1},
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
};

void setup() {
  pinMode(R1, OUTPUT);
  pinMode(R2, OUTPUT);
  pinMode(R3, OUTPUT);
  pinMode(R4, OUTPUT);
  pinMode(R5, OUTPUT);
  pinMode(R6, OUTPUT);
```

```
   pinMode(R7, OUTPUT);
   pinMode(R8, OUTPUT);
   pinMode(C1, OUTPUT);
   pinMode(C2, OUTPUT);
   pinMode(C3, OUTPUT);
   pinMode(C4, OUTPUT);
   pinMode(C5, OUTPUT);
   pinMode(C6, OUTPUT);
   pinMode(C7, OUTPUT);
   pinMode(C8, OUTPUT);
}

void loop() {
   for (int x = 0; x < 5; x++) {
      for (int fila = 0; fila < 8; fila++) {
         digitalWrite(R1 + fila, HIGH);
         digitalWrite(C1, PRUEBA[fila][(0 + position) % 31]);
         digitalWrite(C2, PRUEBA[fila][(1 + position) % 31]);
         digitalWrite(C3, PRUEBA[fila][(2 + position) % 31]);
         digitalWrite(C4, PRUEBA[fila][(3 + position) % 31]);
         digitalWrite(C5, PRUEBA[fila][(4 + position) % 31]);
         digitalWrite(C6, PRUEBA[fila][(5 + position) % 31]);
         digitalWrite(C7, PRUEBA[fila][(6 + position) % 31]);
         digitalWrite(C8, PRUEBA[fila][(7 + position) % 31]);
         delay(2);
         digitalWrite(R1, LOW);
         digitalWrite(R2, LOW);
         digitalWrite(R3, LOW);
         digitalWrite(R4, LOW);
         digitalWrite(R5, LOW);
         digitalWrite(R6, LOW);
         digitalWrite(R7, LOW);
         digitalWrite(R8, LOW);
      }
   }
   position = position + 1;
}
```

Sesión 4. *Actividad 1*

Esquema

Mismo de esquema 10: texto deslizante.

Código

```
1. #include <LiquidCrystal.h>
2.
3. const int rs = 11,
4. en = 12,
5. d4 = 4,
6. d5 = 5,
7. d6 = 6,
8. d7 = 7;
9.
10. LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
11. String Texto = "FRASE";
12.
13. void setup() {
14.    lcd.begin(16, 2);
15. }
16.
17. void loop() {
18.    lcd.setCursor(0,0);P
19.    imprimirD();
20.    for (int positionCounter = 0; positionCounter < (Texto.length() + 16);
       positionCounter++) {
21.       lcd.scrollDisplayRight();
22.       delay(500);
23.    }
24.    lcd.clear();
25.    lcd.setCursor(16,1);
26.    lcd.print(Texto);
27.    for (int positionCounter = 0; positionCounter < (Texto.length() + 16);
       positionCounter++) {
28.       lcd.scrollDisplayLeft();
29.       delay(500);
30.    }
31.    delay(500);
32. }
33.
34. void imprimirD (){
35.    for (int positionCounter = 1; positionCounter <= (Texto.length());
       positionCounter++) {
36.       lcd.clear();
37.       lcd.print(Texto.substring(Texto.length()-positionCounter));
38.       delay(500);
39.    }
40. }
```

Sesión 4. *Actividad 2*

Esquema

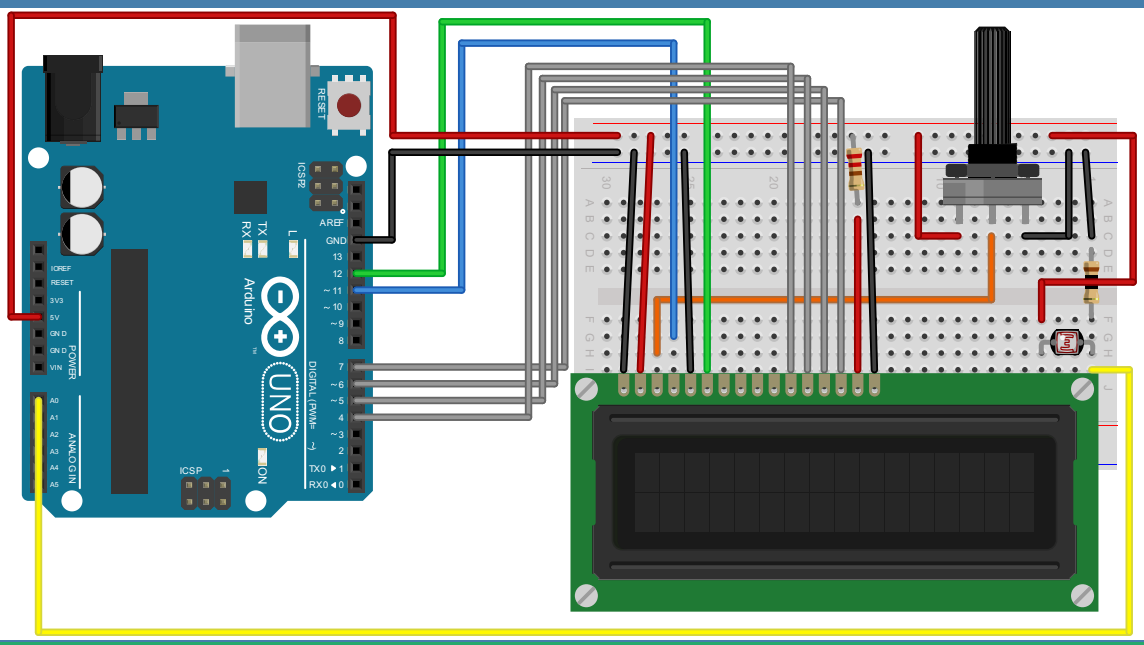
Mismo de esquema 10: texto deslizante.

Código

```
#include <LiquidCrystal.h>

const int rs = 11,
en = 12,
d4 = 4,
d5 = 5,
d6 = 6,
d7 = 7;

int veces = 1;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
String Texto = "3x+2=0";

void setup() {
   lcd.begin(16, 2);
   lcd.setCursor(0,0);
}

void loop() {
   if (veces==1){
   for (int positionCounter = 0; positionCounter <= (Texto.length());
   positionCounter++) {
      lcd.print(Texto.substring(positionCounter,positionCounter+1));
      delay(1000);
   }
   veces += 1;
   }
}
```

Sesión 5. *Actividad 1*

Esquema

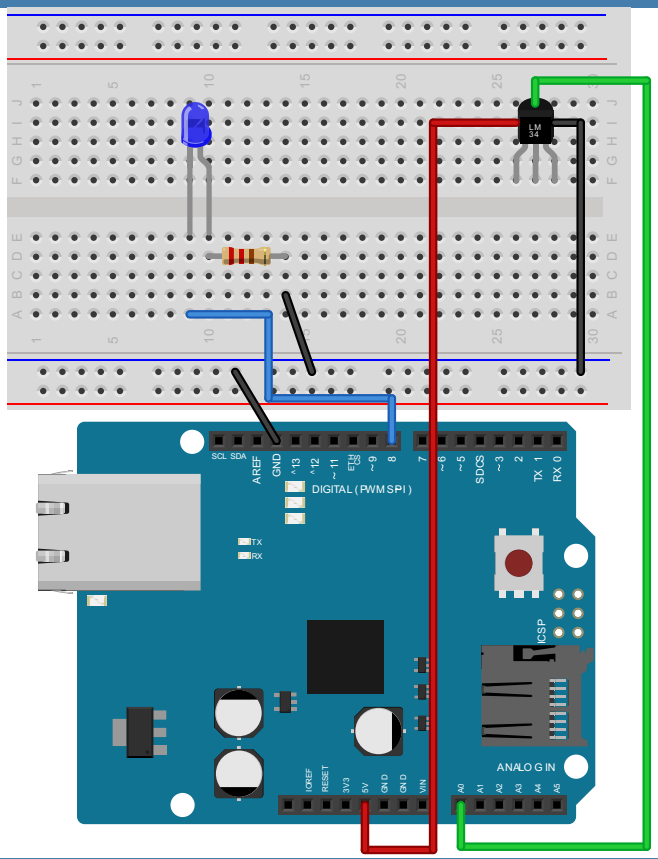


Código

```
#include <LiquidCrystal.h>

int photocellPin = A0;
const int rs = 11,
en = 12,
d4 = 4,
d5 = 5,
d6 = 6,
d7 = 7;
int veces = 1;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
String Texto = "3x+2=0";
int photocellReading;

void setup() {
   lcd.begin(16, 2);
   lcd.print("Temperatura:");
}

void loop() {
   lcd.setCursor(0,1);
   photocellReading = analogRead(photocellPin);
   lcd.println(photocellReading);
   delay(1000);
}
```

Sesión 5. *Actividad 2*

Esquema

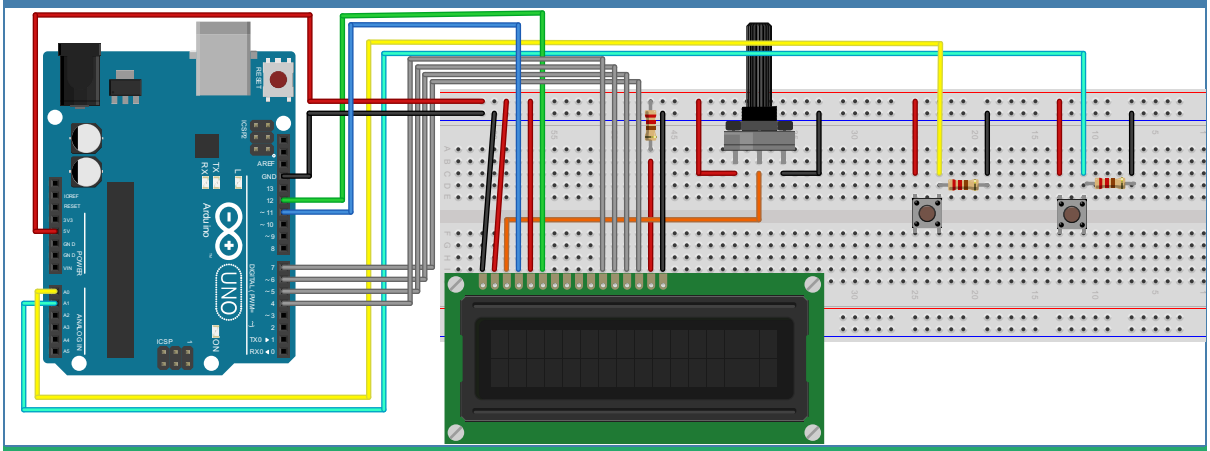


Código

```
int valorTEMP = 0;
int pinLed1 = 8;
int TEMP = A0;

void setup() {
   pinMode(pinLed1, OUTPUT);
}

void loop() {
   valorTEMP = analogRead(TEMP);
   valorTEMP = (5.0 * valorTEMP * 100.0) / 1024.0;
   if (valorTEMP < 21) {
      digitalWrite(pinLed1, HIGH);
   } else if (valorTEMP >= 21) {
      digitalWrite(pinLed1, LOW);
   }
}
```

Sesión 5. *Actividad 3*

Esquema



Código

```
#include <LiquidCrystal.h>

const int rs = 11, en = 12, d4 = 4, d5 = 5, d6 = 6, d7 = 7;

int veces = 1;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
String Texto = "3x+2=0";
const int BOTONR = A0;
int valR;
const int BOTONA = A1;
int valA;

void setup() {
   lcd.begin(16, 2);
   lcd.print("Boton:");
}

void loop() {
   valR = digitalRead(BOTONR);
   valA = digitalRead(BOTONA);
   if (valR==HIGH){
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Boton:");
      lcd.setCursor(0,1);
      lcd.println("ROJO");
   } else if (valA==HIGH){
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Boton:");
      lcd.setCursor(0,1);
      lcd.println("AMARILLO");
   }
   delay(500);
}
```

Sesión 6. *Actividad 1*

Esquema

Mismo de esquema 18: contador ascendente display 7-segmentos.

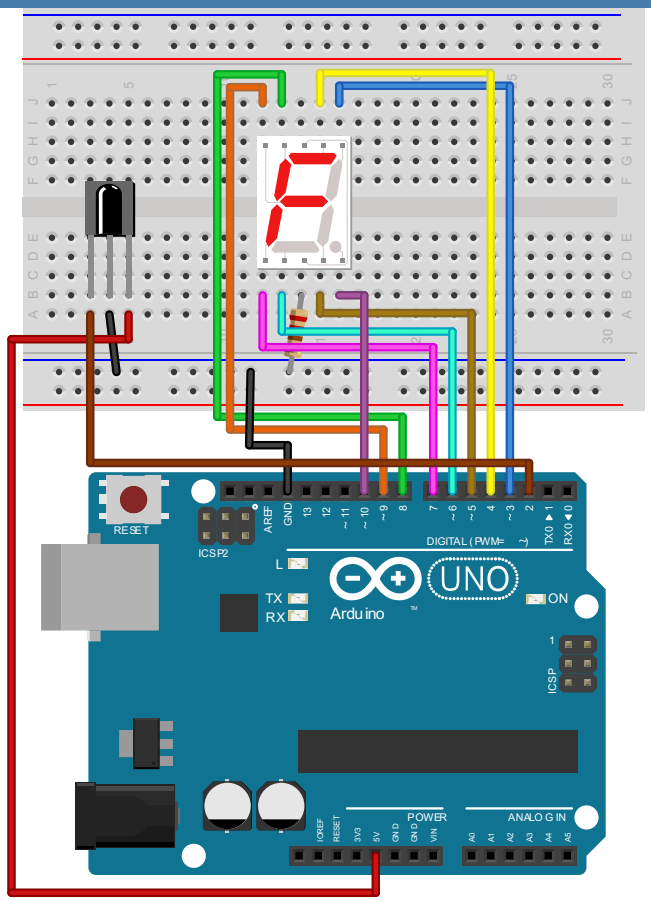
Código

```
long n = 0;
int x = 100;
int del = 1000;
int cont = 0;

byte numero[10][8] = {     { 0, 0, 0, 0, 0, 0, 1, 1},
{ 1, 0, 0, 1, 1, 1, 1, 1},
{ 0, 0, 1, 0, 0, 1, 0, 1},
{ 0, 0, 0, 0, 1, 1, 0, 1},
{ 1, 0, 0, 1, 1, 0, 0, 1},
{ 0, 1, 0, 0, 1, 0, 0, 1},
{ 0, 1, 0, 0, 0, 0, 0, 1},
{ 0, 0, 0, 1, 1, 1, 1, 1},
{ 0, 0, 0, 0, 0, 0, 0, 1},
{ 0, 0, 0, 0, 1, 0, 0, 1}
};

byte grupo [4][4] = {
{ 1, 0, 0, 0,},
{ 0, 1, 0, 0,},
{ 0, 0, 1, 0,},
{ 0, 0, 0, 1,}
};

void setup() {
   for (int i = 2; i < 14; i++) {
      pinMode(i, OUTPUT);
   }
   pinMode(A0, INPUT);
}

void loop(){
   if((n/x)%2==0){
      int t=analogRead(A0);
      limpiar();
      digito(0);
      for (int e = 0; e < 8; e++) {
        digitalWrite(e + 2, numero[(n/x/1000)%10][e]);
      }
      delayMicroseconds(t);
      limpiar();
      digito(1);
      for (int e = 0; e < 8; e++) {
        digitalWrite(e + 2, numero[(n/x/100)%10][e]);
      }
      delayMicroseconds(t);
      limpiar();
      digito(2);
      for (int e = 0; e < 8; e++) {
```

```
50.         digitalWrite(e + 2, numero[(n/x/10)%10][e]);
51.       }
52.       delayMicroseconds(t);
53.       limpiar();
54.       digito(3);
55.       for (int e = 0; e < 8; e++) {
56.         digitalWrite(e + 2, numero[n/x % 10][e]);
57.       }
58.       delayMicroseconds(t);
59.    }
60.    if(n/x > 9999){
61.       n = 0;
62.       delay(5000);
63.    } else n++;
64. }
65.
66. void digito(int x){
67.    for (int e = 0; e < 4; e++) {
68.       digitalWrite(e + 10, grupo[x][e]);
69.    }
70. }
71.
72. void limpiar(){
73.    for (int i = 2; i < 10; i++) {
74.       digitalWrite(i, HIGH);
75.    }
76. }
```

Sesión 6. *Actividad 2*

Esquema



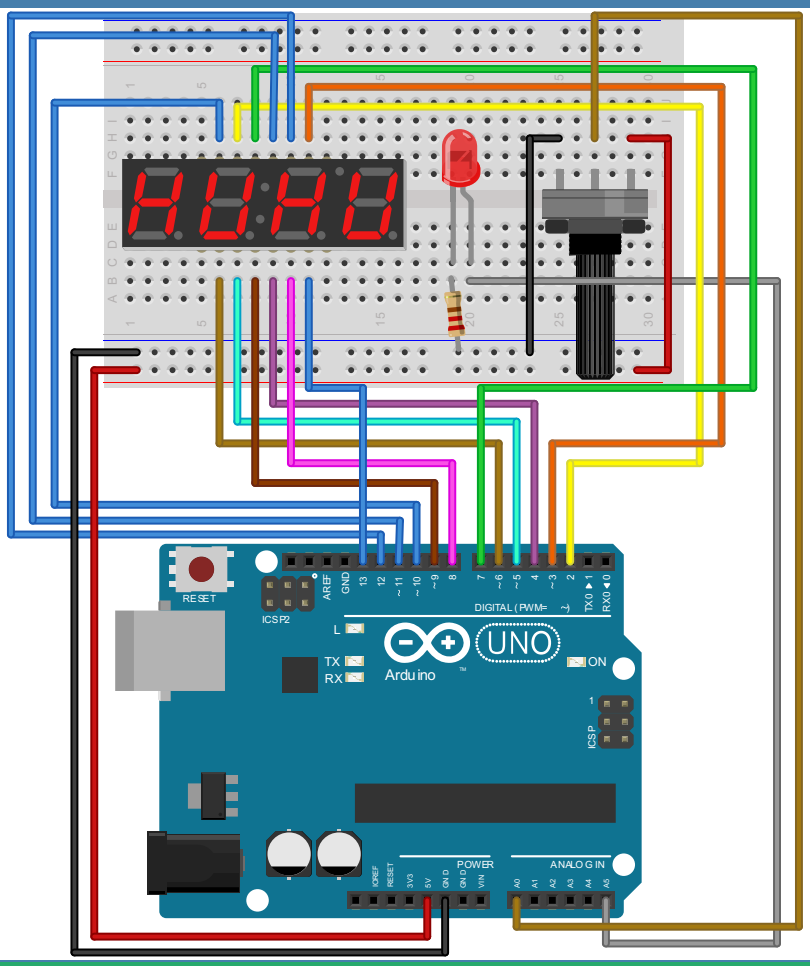
Código

```
#include <IRremote.h>

int RECV_PIN = 2;
IRrecv irrecv(RECV_PIN);
decode_results results;
unsigned long key_value = 0;

byte numero[10][8] = {
{ 1, 1, 1, 1, 1, 1, 0, 0 },
{ 1, 0, 1, 0, 0, 0, 0, 0 },
{ 1, 1, 0, 1, 1, 0, 1, 0 },
{ 1, 1, 1, 1, 0, 0, 1, 0 },
{ 1, 0, 1, 0, 0, 1, 1, 0 },
{ 0, 1, 1, 1, 0, 1, 1, 0 },
{ 0, 1, 1, 1, 1, 1, 1, 0 },
{ 1, 1, 1, 0, 0, 0, 0, 0 },
{ 1, 1, 1, 1, 1, 1, 1, 0 },
{ 1, 1, 1, 1, 0, 1, 1, 0 }
};

void setup() {
   Serial.begin(9600);
   for (int i = 3; i < 10; i++) {
```

```
      pinMode(i, OUTPUT);
   }
   irrecv.enableIRIn();
   irrecv.blink13(true);
}

void loop() {
   if (irrecv.decode(&results)) {
      switch (results.value) {
         case 0xFF6897:
         for (int e = 0; e < 8; e++) {
            digitalWrite(e + 3, numero[0][e]);  }
         break;
         case 0xFF30CF:
         for (int e = 0; e < 8; e++) {
            digitalWrite(e + 3, numero[1][e]);  }
         break;
         case 0xFF18E7:
         for (int e = 0; e < 8; e++) {
            digitalWrite(e + 3, numero[2][e]);  }
         break;
         case 0xFF7A85:
         for (int e = 0; e < 8; e++) {
            digitalWrite(e + 3, numero[3][e]);  }
         break;
         case 0xFF10EF:
         for (int e = 0; e < 8; e++) {
            digitalWrite(e + 3, numero[4][e]);  }
         break;
         case 0xFF38C7:
         for (int e = 0; e < 8; e++) {
            digitalWrite(e + 3, numero[5][e]);  }
         break;
         case 0xFF5AA5:
         for (int e = 0; e < 8; e++) {
            digitalWrite(e + 3, numero[6][e]);  }
         break;
         case 0xFF42BD:
         for (int e = 0; e < 8; e++) {
            digitalWrite(e + 3, numero[7][e]);  }
         break;
         case 0xFF4AB5:
         for (int e = 0; e < 8; e++) {
            digitalWrite(e + 3, numero[8][e]);  }
         break;
         case 0xFF52AD:
         for (int e = 0; e < 8; e++) {
            digitalWrite(e + 3, numero[9][e]);  }
         break;
      }
      delay(500);
      key_value = results.value;
      irrecv.resume();
   }
}
```

Sesión 6. *Actividad 3*

Esquema



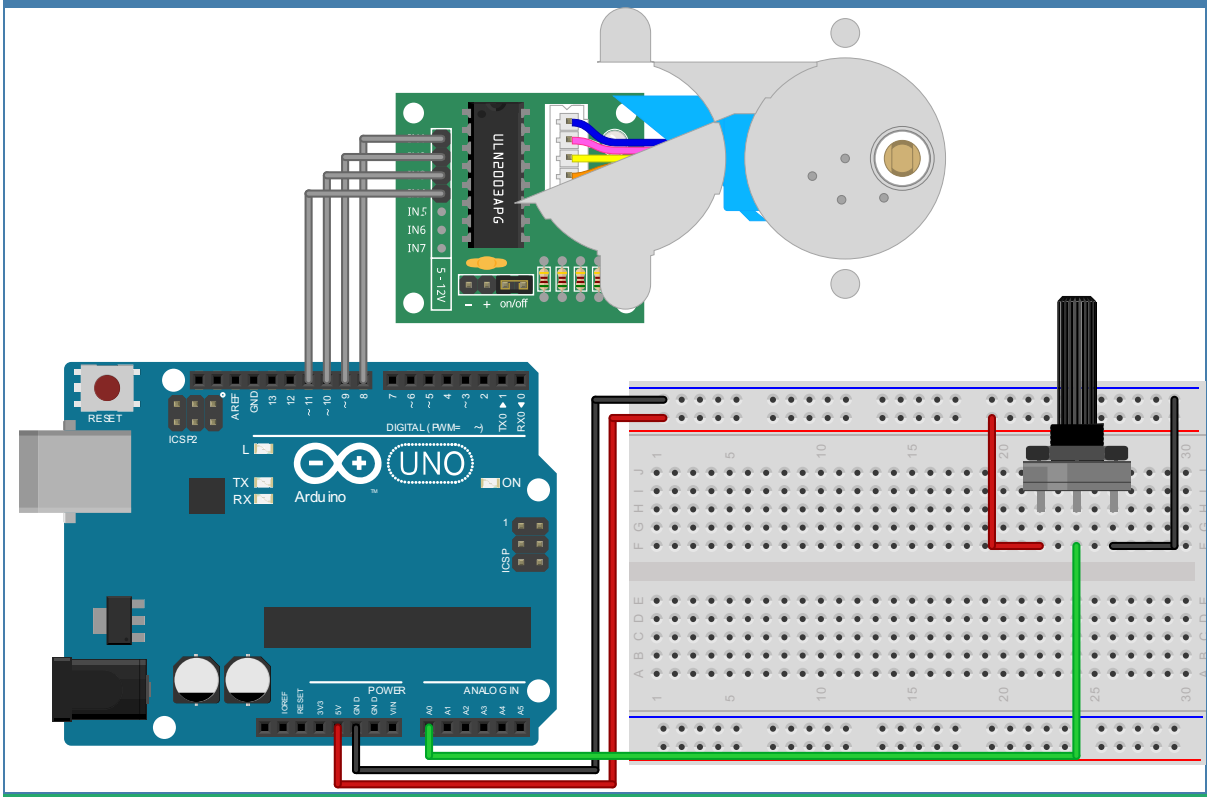
Código

```
long n = 100100;
int x = 100;
const int LED=A5;
int cont = 0;

byte numero[10][8] = {   { 0, 0, 0, 0, 0, 0, 1, 1},
{ 1, 0, 0, 1, 1, 1, 1, 1},
{ 0, 0, 1, 0, 0, 1, 0, 1},
{ 0, 0, 0, 0, 1, 1, 0, 1},
{ 1, 0, 0, 1, 1, 0, 0, 1},
{ 0, 1, 0, 0, 1, 0, 0, 1},
{ 0, 1, 0, 0, 0, 0, 0, 1},
{ 0, 0, 0, 1, 1, 1, 1, 1},
{ 0, 0, 0, 0, 0, 0, 0, 1},
{ 0, 0, 0, 0, 1, 0, 0, 1}
};

byte grupo [4][4] = {
{ 1, 0, 0, 0,},
```

```
{ 0, 1, 0, 0,},
{ 0, 0, 1, 0,},
{ 0, 0, 0, 1,}
};

void setup() {
   for (int i = 2; i < 14; i++) {
      pinMode(i, OUTPUT);  }
   pinMode(A0, INPUT);
   pinMode(LED,OUTPUT);
}

void loop(){
   int t=analogRead(A0);
   limpiar();
   digito(0);
   for (int e = 0; e < 8; e++) {
      digitalWrite(e + 2, numero[(n/x/1000)%10][e]);  }
   delayMicroseconds(t);
   limpiar();
   digito(1);
   for (int e = 0; e < 8; e++) {
      digitalWrite(e + 2, numero[(n/x/100)%10][e]);    }
   delayMicroseconds(t);
   limpiar();
   digito(2);
   for (int e = 0; e < 8; e++) {
      digitalWrite(e + 2, numero[(n/x/10)%10][e]);  }
   delayMicroseconds(t);
   limpiar();
   digito(3);
   for (int e = 0; e < 8; e++) {
      digitalWrite(e + 2, numero[n/x % 10][e]);  }
   delayMicroseconds(t);
   if ((n/x)%11==0){
      digitalWrite(LED,HIGH);
   } else {
   digitalWrite(LED,LOW);}
   if(n/x < 1){
      n = 100100;
      delay(5000);
   } else n--;  }

void digito(int x){
   for (int e = 0; e < 4; e++) {
      digitalWrite(e + 10, grupo[x][e]);  }
}
void limpiar(){
   for (int i = 2; i < 10; i++) {
      digitalWrite( i, HIGH);  }
}
```

Sesión 7. *Actividad 1*

Esquema



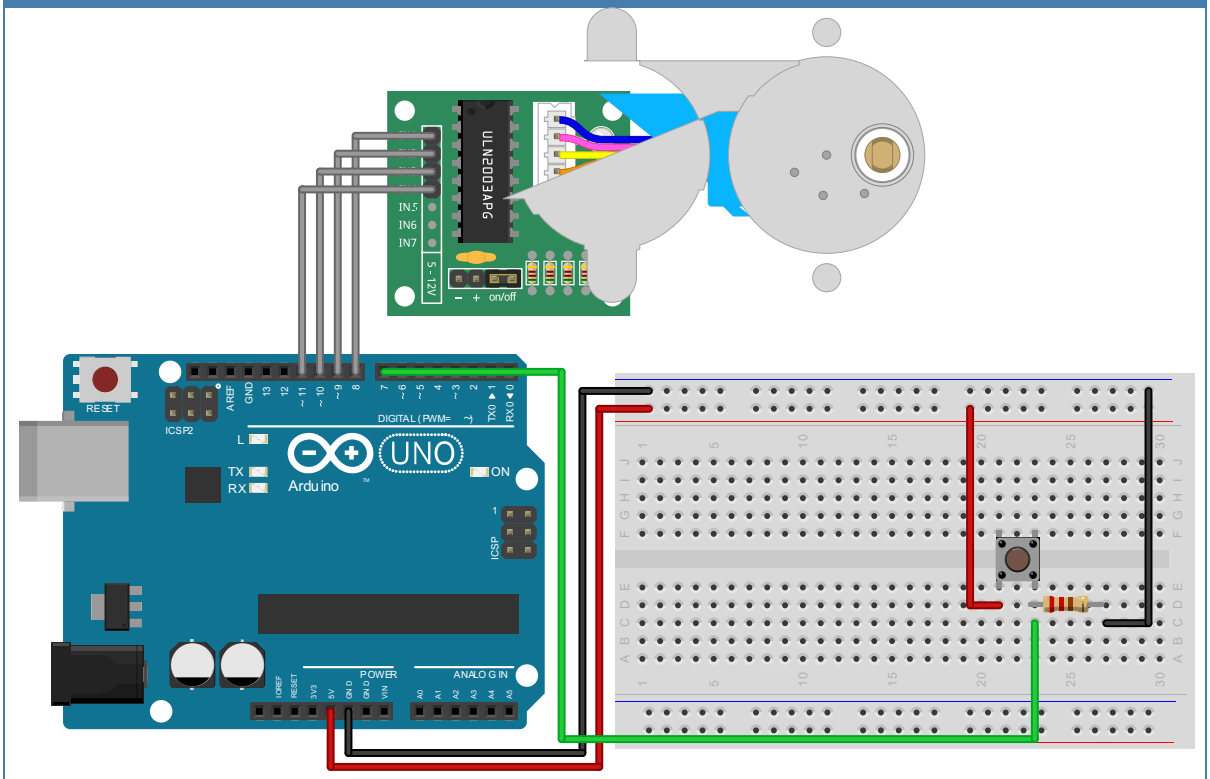
Código

```
1.  int IN1 = 8;
2.  int IN2 = 9;
3.  int IN3 = 10;
4.  int IN4 = 11;
5.  int demora = 0;
6.
7.  void setup() {
8.     pinMode (IN1, OUTPUT);
9.     pinMode (IN2, OUTPUT);
10.    pinMode (IN3, OUTPUT);
11.    pinMode (IN4, OUTPUT);
12.    pinMode(A0, INPUT);
13. }
14.
15. void loop(){
16.    for (int i = 0; i < 512; i++)  {
17.       demora=analogRead(A0);
18.       digitalWrite (IN1, HIGH);
19.       digitalWrite (IN2, LOW);
20.       digitalWrite (IN3, LOW);
21.       digitalWrite (IN4, LOW);
22.       delay(demora);
23.       digitalWrite (IN1, LOW);
24.       digitalWrite (IN2, HIGH);
```

```
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, LOW);
    delay(demora);
    digitalWrite (IN1, LOW);
    digitalWrite (IN2, LOW);
    digitalWrite (IN3, HIGH);
    digitalWrite (IN4, LOW);
    delay(demora);
    digitalWrite (IN1, LOW);
    digitalWrite (IN2, LOW);
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, HIGH);
    delay(demora);
  }
  digitalWrite (IN1, LOW);
  digitalWrite (IN2, LOW);
  digitalWrite (IN3, LOW);
  digitalWrite (IN4, LOW);
  delay(2000);
}
```

Sesión 7. *Actividad 2*

Esquema



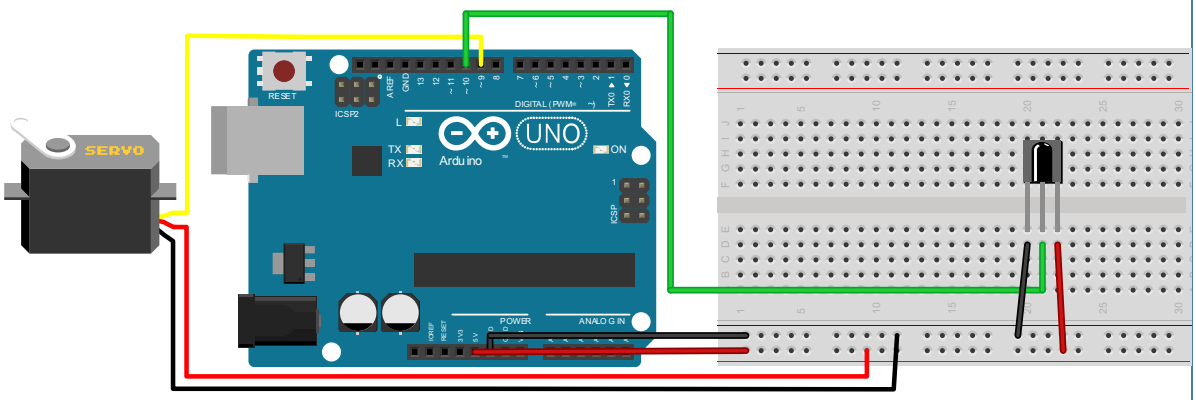
Código

```
int IN1 = 8;
int IN2 = 9;
int IN3 = 10;
int IN4 = 11;
int BOTON = 7;
int demora = 20;
int detener = 0;

void setup() {
   pinMode (IN1, OUTPUT);
   pinMode (IN2, OUTPUT);
   pinMode (IN3, OUTPUT);
   pinMode (IN4, OUTPUT);
   pinMode(BOTON, INPUT);
}

void loop(){
   if (detener==0){
      for (int i = 0; i < 512; i++) {
         digitalWrite (IN1, HIGH);
         digitalWrite (IN2, LOW);
         digitalWrite (IN3, LOW);
         digitalWrite (IN4, LOW);
```

```
24.         delay(demora);
25.         digitalWrite (IN1, LOW);
26.         digitalWrite (IN2, HIGH);
27.         digitalWrite (IN3, LOW);
28.         digitalWrite (IN4, LOW);
29.         delay(demora);
30.         digitalWrite (IN1, LOW);
31.         digitalWrite (IN2, LOW);
32.         digitalWrite (IN3, HIGH);
33.         digitalWrite (IN4, LOW);
34.         delay(demora);
35.         digitalWrite (IN1, LOW);
36.         digitalWrite (IN2, LOW);
37.         digitalWrite (IN3, LOW);
38.         digitalWrite (IN4, HIGH);
39.         delay(demora);
40.         if (digitalRead(BOTON)==LOW){
41.           detener=1;
42.         }
43.       }
44.       digitalWrite (IN1, LOW);
45.       digitalWrite (IN2, LOW);
46.       digitalWrite (IN3, LOW);
47.       digitalWrite (IN4, LOW);
48.       if (digitalRead(BOTON)==LOW){
49.         detener=1;
50.       }
51.       delay(2000);
52.    }
53. }
```

Sesión 7. *Actividad 3*

Esquema



Código

```
#include <Servo.h>
#include <IRremote.h>

int RECV_PIN = 10;
IRrecv irrecv(RECV_PIN);
decode_results  results;
unsigned long key_value=0;
Servo myservo;
int angulo = 0 ;

void setup() {
   myservo.attach(9);
   irrecv.enableIRIn();
   irrecv.blink13(true);
}

void loop() {
   if (irrecv.decode(&results))  {
      irrecv.resume();
      if (results.value == 0xFFFFFFFF){
         results.value = key_value;}
      switch (results.value){
         case 0xFF6897:
         if (angulo == 0) {
            angulo = 0;
         } else {
            angulo = angulo*10 + 0;    }
         break;
         case 0xFF30CF:
         if (angulo == 0) {
            angulo = 1;
         } else {
            angulo = angulo*10 + 1;    }
         break;
         case 0xFF18E7:
         if (angulo == 0) {
            angulo = 2;
```

```
        } else {
           angulo = angulo*10 + 2;   }
        break;
        case 0xFF7A85:
        if (angulo == 0) {
           angulo = 3;
        } else {
           angulo = angulo*10 + 3;   }
        break;
        case 0xFF10EF:
        if (angulo == 0) {
           angulo = 4;
        } else {
           angulo = angulo*10 + 4;   }
        break;
        case 0xFF38C7:
        if (angulo == 0) {
           angulo = 5;
        } else {
           angulo = angulo*10 + 5;   }
        break;
        case 0xFF5AA5:
        if (angulo == 0) {
           angulo = 6;
        } else {
           angulo = angulo*10 + 6;   }
        break;
        case 0xFF42BD:
        if (angulo == 0) {
           angulo = 7;
        } else {
           angulo = angulo*10 + 7;   }
        break;
        case 0xFF4AB5:
        if (angulo == 0) {
           angulo = 8;
        } else {
           angulo = angulo*10 + 8;   }
        break;
        case 0xFF52AD:
        if (angulo == 0) {
           angulo = 9;
        } else {
           angulo = angulo*10 + 9;   }
        break;
        case 0xFF906F:
        myservo.write(angulo);
        angulo = 0;
        break;
        delay(10);
      }
   }
}
```

Anexos

Anexo 1: Rúbrica para evaluar las sesiones de trabajo

Criterio	Excelente (4)	Bueno (3)	Aceptable (2)	Insuficiente (1)
Comprensión del Concepto	Demuestra una comprensión completa y profunda de los conceptos y principios involucrados. Puede explicar y aplicar los conceptos sin errores.	Demuestra una comprensión buena de los conceptos, con solo algunos errores menores. Puede aplicar los conceptos correctamente.	Demuestra una comprensión básica de los conceptos, pero con varios errores. La aplicación de los conceptos es limitada.	Demuestra poca o ninguna comprensión de los conceptos. Incapaz de aplicar los conceptos correctamente.
Ejecución Técnica	Completa las tareas con precisión y sin errores. Realiza todas las conexiones y programas de manera correcta y eficiente.	Completa las tareas con precisión, aunque con algunos errores menores. La mayoría de las conexiones y programas son correctos.	Completa las tareas, pero con varios errores. Algunas conexiones y programas funcionan, pero otros no.	No completa las tareas correctamente. La mayoría de las conexiones y programas contienen errores significativos.
Solución de Problemas	Identifica y soluciona problemas de manera independiente y eficaz. Muestra creatividad y pensamiento crítico en la resolución de problemas.	Identifica y soluciona problemas con poca ayuda. Muestra algo de creatividad y pensamiento crítico en la resolución de problemas.	Necesita ayuda significativa para identificar y solucionar problemas. Muestra poca creatividad o pensamiento crítico.	Incapaz de identificar y solucionar problemas, incluso con ayuda. No muestra creatividad ni pensamiento crítico.
Documentación y Presentación	Documenta todos los pasos de manera clara, detallada y organizada. Presenta su trabajo de forma profesional y comprensible.	Documenta la mayoría de los pasos de manera clara y organizada. La presentación es comprensible, pero podría mejorarse en algunos aspectos.	Documenta algunos pasos, pero de manera incompleta o desorganizada. La presentación tiene varios aspectos que necesitan mejora.	No documenta adecuadamente los pasos. La presentación es desorganizada y difícil de entender.
Trabajo en Equipo y Colaboración	Colabora de manera efectiva con sus compañeros. Contribuye significativamente al trabajo del equipo y respeta las	Colabora bien con sus compañeros. Contribuye al trabajo del equipo y respeta las opiniones de los demás, aunque	Colabora de manera limitada con sus compañeros. Contribuye poco al trabajo del equipo y a veces	No colabora con sus compañeros. No contribuye al trabajo del equipo y no respeta las

	opiniones de los demás.	con algunas dificultades.	tiene dificultades para respetar las opiniones de los demás.	opiniones de los demás.
Reflexión y Autoevaluación	Reflexiona profundamente sobre su aprendizaje y desempeño. Identifica claramente fortalezas y áreas de mejora.	Reflexiona sobre su aprendizaje y desempeño, aunque de manera superficial. Identifica algunas fortalezas y áreas de mejora.	Reflexiona de manera limitada sobre su aprendizaje y desempeño. Identifica pocas fortalezas y áreas de mejora.	No reflexiona sobre su aprendizaje y desempeño. No identifica fortalezas ni áreas de mejora.