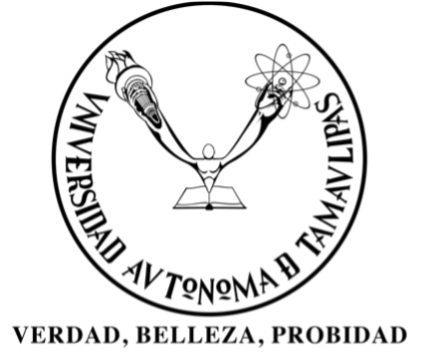
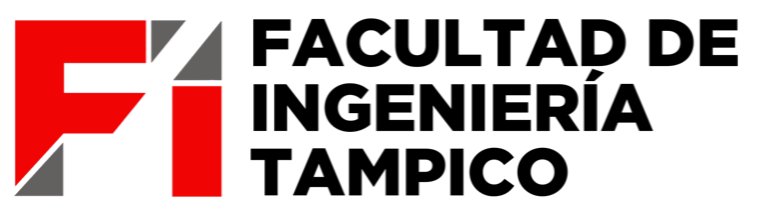
**UNIVERSIDAD AUTÓNOMA DE TAMAULIPAS**

**Facultad de Ingeniería Tampico**

**PRÁCTICAS**

****

**Unidad 1**

**Diseño Electrónico Basado en Sistemas Embebidos I**

Docente: **Dr. García Ruiz Alejandro Humberto**

8vo. Semestre – Grupo “I”

2024-1

**Fecha de Entrega:** 19 de mayo de 2024

**Integrante(s):**

Aros Charles Ángel Alberto- a2193330129

Hernandez Balleza Jose Eduardo- a2163226011

Moreno Wilches Fernando- a2193330272

**ÍNDICE**

[1. Actividades de Clase 3](#_Toc158809433)

[2. Tareas e Investigaciones 3](#_Toc158809434)

[3. Programas 3](#_Toc158809435)

[4. Prácticas / Proyectos 3](#_Toc158809436)

# 1. Actividades de Clase

EJERCICIO 1.

EJERCICIO 2.

# 2. Tareas e Investigaciones

**Sistema Electrónico:**

Un sistema electrónico es un entorno complejo de hardware y software diseñado para realizar tareas específicas mediante el procesamiento de señales eléctricas. Estos sistemas están compuestos por una variedad de componentes electrónicos interconectados que trabajan juntos de manera coordinada para cumplir con sus funciones designadas. Desde los dispositivos más simples hasta los sistemas más avanzados, la esencia sigue siendo la misma: la manipulación controlada de señales eléctricas para lograr una tarea determinada.

Para entender mejor cómo funciona un sistema electrónico, es útil desglosarlo en sus componentes básicos:

1. Componentes Pasivos: Estos son elementos básicos del circuito que no amplifican ni controlan la corriente eléctrica, pero son esenciales para el funcionamiento del sistema. Esto incluye resistencias, condensadores, inductores y diodos. Por ejemplo, las resistencias limitan el flujo de corriente, los condensadores almacenan energía eléctrica y los diodos permiten que la corriente fluya en una dirección específica.

2. Componentes Activos: Estos son los componentes que pueden controlar la corriente eléctrica, amplificar señales o realizar operaciones lógicas. Los ejemplos incluyen transistores, circuitos integrados, microcontroladores y procesadores. Los transistores, por ejemplo, pueden actuar como interruptores o amplificadores de señal, mientras que los microcontroladores y procesadores ejecutan instrucciones y procesan datos.

3. Dispositivos de Entrada y Salida (E/S): Estos componentes permiten la interacción del sistema electrónico con el entorno externo. Los dispositivos de entrada, como sensores y teclados, recopilan datos del entorno. Los dispositivos de salida, como pantallas y actuadores, presentan información al usuario o realizan acciones en respuesta a las entradas del usuario.

4. Interconexiones y Circuitos: Estas son las rutas físicas y lógicas que conectan todos los componentes del sistema. Los circuitos pueden ser simples o complejos, dependiendo de la función que el sistema deba realizar. La disposición y el diseño de estos circuitos son críticos para el funcionamiento correcto del sistema electrónico.

5. Software y Firmware: Junto con el hardware, el software y el firmware desempeñan un papel crucial en la operación de un sistema electrónico. El software controla el funcionamiento del sistema, ejecutando programas y procesos para lograr sus objetivos. El firmware es un tipo de software integrado en componentes específicos del hardware, como microcontroladores, para controlar su funcionamiento.

**Sistema Embebido:**

Un sistema embebido, también conocido como sistema integrado o sistema empotrado, es un sistema informático diseñado para realizar una o varias funciones específicas dentro de un dispositivo más grande o sistema, en contraposición a un sistema de propósito general, como una computadora personal. Estos sistemas están integrados en el dispositivo que controlan y están diseñados típicamente para cumplir con una tarea o un conjunto limitado de tareas de manera eficiente y específica.

Los sistemas embebidos se encuentran en una amplia gama de dispositivos y aplicaciones en la vida cotidiana, desde electrodomésticos como hornos de microondas y lavadoras hasta dispositivos médicos, automóviles, sistemas de control industrial, juguetes electrónicos y más. Pueden ser tan simples como un microcontrolador que controla un sistema de apertura de puertas o tan complejos como un sistema de navegación y entretenimiento en un automóvil.

Algunas características importantes de los sistemas embebidos incluyen:

1. Hardware Específico: Los sistemas embebidos están diseñados con hardware específico para la tarea que realizan. Esto puede incluir microcontroladores, microprocesadores especializados, circuitos integrados personalizados, sensores y actuadores específicos, entre otros componentes.

2. Software Optimizado: El software que ejecuta un sistema embebido está altamente optimizado para la tarea que debe realizar. Puede ser un sistema operativo en tiempo real (RTOS) o un software desarrollado específicamente para la aplicación, con un tamaño y una complejidad adaptados a los recursos limitados del hardware.

3. Consumo de Energía Eficiente: Dado que muchos sistemas embebidos funcionan con baterías o tienen limitaciones de energía, se diseñan para ser altamente eficientes en el consumo de energía. Esto implica la optimización tanto del hardware como del software para minimizar el uso de energía cuando el sistema está inactivo o en modo de bajo consumo.

4. Tamaño Compacto: Los sistemas embebidos suelen ser compactos en tamaño y pueden estar integrados en el diseño físico del dispositivo que controlan. Esto los hace ideales para dispositivos donde el espacio es limitado o donde se necesita una integración física estrecha con otros componentes.

En resumen, un sistema embebido es un sistema informático diseñado para realizar funciones específicas dentro de un dispositivo más grande o sistema. Estos sistemas están integrados en el dispositivo que controlan y están diseñados para ser eficientes en energía, compactos en tamaño y altamente especializados para la tarea que realizan.

**Automatización e Inteligencia:**

La automatización se refiere al proceso de realizar tareas o procesos de manera automática, sin intervención humana directa. Esto se logra mediante el uso de sistemas mecánicos, electrónicos, informáticos y/o robóticos que ejecutan acciones programadas para llevar a cabo una tarea específica. La automatización puede aplicarse en una amplia gama de áreas, desde la fabricación industrial y la logística hasta la domótica en el hogar y los sistemas de información empresarial.

Por otro lado, la inteligencia, en el contexto de la tecnología, puede referirse a la capacidad de un sistema para adquirir, comprender, aplicar conocimientos y adaptarse a nuevas situaciones. La inteligencia artificial (IA) es un campo de la informática que se enfoca en el desarrollo de sistemas capaces de simular el comportamiento inteligente humano. Estos sistemas pueden incluir algoritmos de aprendizaje automático, redes neuronales artificiales y otras técnicas para procesar datos, reconocer patrones, tomar decisiones y aprender de la experiencia.

Cuando se combinan la automatización y la inteligencia, se pueden crear sistemas automatizados más avanzados y sofisticados que no solo ejecutan tareas repetitivas de manera eficiente, sino que también son capaces de adaptarse, aprender y tomar decisiones autónomas en función de las condiciones cambiantes del entorno. Por ejemplo, en la industria, la combinación de la automatización con la inteligencia artificial puede permitir la optimización de procesos de fabricación, la predicción de fallos de maquinaria y la personalización de la producción para satisfacer las demandas del mercado de manera más eficiente. En resumen, la automatización e inteligencia trabajan juntas para crear sistemas más eficientes, adaptables y autónomos en una variedad de aplicaciones y sectores.

**Domótica e Inmótica:**

La domótica y la inmótica son dos conceptos relacionados pero que se aplican en contextos diferentes:

1. Domótica: La domótica se refiere al conjunto de tecnologías aplicadas al control y automatización inteligente de la vivienda y el hogar. Su objetivo es mejorar la calidad de vida, la seguridad, el confort y la eficiencia energética en el entorno residencial. La domótica permite la gestión centralizada y remota de dispositivos y sistemas dentro del hogar, como iluminación, climatización, sistemas de seguridad, electrodomésticos, entre otros. Los sistemas domóticos suelen controlarse mediante interfaces como aplicaciones móviles, paneles de control táctiles o comandos de voz, y pueden adaptarse a las preferencias y necesidades específicas de los usuarios.

2. Inmótica: La inmótica, por otro lado, se refiere a la aplicación de tecnologías similares, pero en el ámbito de los edificios y las instalaciones comerciales o industriales. Su objetivo principal es optimizar la gestión y el control de los sistemas de climatización, iluminación, seguridad, control de accesos, energía, entre otros, en edificios y grandes instalaciones. La inmótica busca mejorar la eficiencia operativa, reducir los costos de mantenimiento y energía, y proporcionar un ambiente más seguro y confortable para los ocupantes del edificio. Los sistemas inmóticos pueden ser más complejos que los sistemas domóticos debido a las dimensiones y las necesidades específicas de los edificios comerciales e industriales.

**Ambientes Inteligentes:**

Los ambientes inteligentes, también conocidos como "entornos inteligentes" o "espacios inteligentes", son entornos físicos que están equipados con tecnología avanzada para interactuar y adaptarse de manera inteligente a las necesidades y preferencias de los usuarios. Estos entornos incorporan una variedad de dispositivos, sensores, sistemas de procesamiento de datos y software de control para crear un espacio que puede monitorizar, analizar y responder a diversas situaciones de manera automatizada y autónoma.

Los ambientes inteligentes pueden encontrarse en una variedad de contextos, incluidos hogares, oficinas, edificios comerciales, hospitales, ciudades inteligentes y más. Algunas características clave de los ambientes inteligentes incluyen:

1. Interconexión de dispositivos: Los dispositivos en un ambiente inteligente están interconectados a través de redes de comunicación, como Wi-Fi, Bluetooth o redes cableadas, permitiendo que compartan datos y coordinen acciones entre sí.

2. Sensores y dispositivos de entrada: Los ambientes inteligentes están equipados con una variedad de sensores, como sensores de movimiento, sensores de temperatura, sensores de luz, micrófonos, cámaras, entre otros, que recopilan información sobre el entorno y las actividades de los usuarios.

3. Análisis de datos y toma de decisiones: Los datos recopilados por los sensores son procesados por sistemas de inteligencia artificial y algoritmos de aprendizaje automático para extraer información significativa y tomar decisiones inteligentes en función de los objetivos del ambiente inteligente y las necesidades de los usuarios.

4. Automatización y control: Los ambientes inteligentes pueden automatizar una variedad de tareas y procesos, como el control de la iluminación, la climatización, los sistemas de seguridad, los dispositivos multimedia, entre otros, según las preferencias y los patrones de uso de los usuarios.

5. Adaptabilidad y personalización: Los ambientes inteligentes pueden adaptarse dinámicamente a las necesidades y preferencias individuales de los usuarios, proporcionando una experiencia personalizada y optimizada en función de factores como la hora del día, la ubicación, el clima, la actividad del usuario y más.

**Ciudades Inteligentes:**

Las ciudades inteligentes, también conocidas como "smart cities", son áreas urbanas que emplean tecnología de la información y las comunicaciones (TIC) para mejorar la calidad de vida de sus habitantes, gestionar eficientemente los recursos y servicios, y promover el desarrollo sostenible y la innovación. Estas ciudades utilizan una variedad de sensores, dispositivos conectados, redes de comunicación y plataformas de datos para recopilar información en tiempo real sobre diversos aspectos de la vida urbana y tomar decisiones informadas para optimizar el funcionamiento de la ciudad.

Algunas características y componentes clave de las ciudades inteligentes incluyen:

1. Infraestructura tecnológica: Las ciudades inteligentes están equipadas con una infraestructura tecnológica robusta que incluye redes de comunicación de alta velocidad, sensores distribuidos por toda la ciudad, sistemas de gestión de datos y plataformas de análisis para recopilar, procesar y gestionar grandes volúmenes de información.

2. Gestión de recursos y servicios: Las ciudades inteligentes optimizan la gestión de recursos y servicios urbanos, como energía, agua, transporte, residuos, iluminación pública, seguridad y salud pública, mediante el uso de tecnología para monitorear y controlar de manera eficiente el uso y la distribución de estos recursos.

3. Movilidad sostenible: Las ciudades inteligentes promueven la movilidad sostenible mediante la implementación de sistemas de transporte público eficientes, la integración de tecnologías de transporte inteligente, como el transporte público eléctrico y compartido, y la creación de infraestructuras para facilitar el uso de bicicletas y peatones.

4. Participación ciudadana: Las ciudades inteligentes fomentan la participación activa de los ciudadanos en la toma de decisiones y el diseño de políticas urbanas mediante el uso de tecnologías de participación ciudadana, como aplicaciones móviles, plataformas en línea y sistemas de votación electrónica.

5. Desarrollo económico y social: Las ciudades inteligentes impulsan el desarrollo económico y social al fomentar la innovación, el emprendimiento y la creación de empleo en sectores emergentes como la tecnología, la energía limpia y la economía digital.

**Software y hardware para sistemas embebidos:**

Los sistemas embebidos requieren tanto software como hardware específicos para cumplir con sus funciones diseñadas. Aquí hay una descripción de ambos:

1. Software para sistemas embebidos:

El software para sistemas embebidos es el conjunto de programas y algoritmos diseñados para ejecutarse en el hardware embebido. Este software controla las operaciones del sistema, procesa datos, gestiona la interacción con el entorno y realiza las funciones específicas para las que fue diseñado el sistema.

Puede incluir sistemas operativos en tiempo real (RTOS), que son sistemas operativos diseñados para responder a eventos y solicitudes en tiempo real, garantizando tiempos de respuesta predecibles. Ejemplos de RTOS incluyen FreeRTOS, VxWorks, uC/OS-II, entre otros.

También puede incluir software de aplicación específico, desarrollado para cumplir con las necesidades particulares del sistema embebido. Este software puede estar escrito en lenguajes como C, C++, Java, Python, entre otros, y puede ejecutar tareas como controlar dispositivos, recopilar datos, procesar información y comunicarse con otros sistemas.

2. Hardware para sistemas embebidos:

El hardware para sistemas embebidos consiste en los componentes físicos que forman parte del sistema embebido, incluyendo procesadores, memoria, interfaces de entrada/salida (E/S), sensores, actuadores, circuitos integrados y otros dispositivos.

Los procesadores en los sistemas embebidos pueden variar en complejidad, desde microcontroladores simples hasta microprocesadores más potentes. Estos procesadores ejecutan el software embebido y controlan las operaciones del sistema.

La memoria en los sistemas embebidos puede incluir memoria volátil (RAM) para almacenar datos en tiempo de ejecución, así como memoria no volátil (ROM, Flash) para almacenar el software y la configuración del sistema.

Las interfaces de entrada/salida (E/S) permiten la interacción del sistema embebido con el entorno externo, incluyendo puertos seriales, USB, Ethernet, GPIO (General Purpose Input/Output), entre otros.

Los sensores y actuadores son dispositivos utilizados para capturar datos del entorno y realizar acciones físicas, respectivamente. Estos pueden incluir sensores de temperatura, acelerómetros, cámaras, motores, luces, entre otros.

**Ensayo de las tareas anteriores:**

En la era actual de la tecnología, la integración de sistemas embebidos en entornos inteligentes marca un hito significativo en la transformación digital de nuestras vidas. Desde el desarrollo de sistemas electrónicos hasta la implementación de ciudades inteligentes, cada paso refleja un progreso hacia una sociedad más conectada, automatizada e inteligente. Este ensayo profundizará en los diversos aspectos de esta integración, explorando los desafíos, oportunidades y el potencial transformador que presenta para el futuro.

Los sistemas electrónicos, que sirven como los bloques de construcción fundamentales de la tecnología moderna, proporcionan la base sobre la cual se construyen los sistemas embebidos. Estos últimos, caracterizados por su tamaño compacto y su capacidad de realizar funciones específicas, son la columna vertebral de la automatización en una amplia gama de sectores, desde la manufactura hasta la atención médica y la movilidad urbana. La proliferación de sistemas embebidos está estrechamente ligada a la evolución de la electrónica, que ha permitido la miniaturización y la optimización de componentes críticos.

La automatización e inteligencia representan las piedras angulares de la innovación en sistemas embebidos. La automatización permite la ejecución autónoma de tareas previamente realizadas por humanos, lo que conduce a mejoras en la eficiencia, la productividad y la seguridad. Por otro lado, la inteligencia artificial (IA) brinda a los sistemas embebidos la capacidad de aprender de datos históricos, adaptarse a nuevas situaciones y tomar decisiones informadas en tiempo real. Esta combinación de automatización e inteligencia está impulsando avances significativos en campos como la robótica, la medicina personalizada y la conducción autónoma.

La domótica e inmótica ilustran cómo los sistemas embebidos están transformando nuestros hogares y espacios comerciales. La domótica, que se enfoca en la automatización inteligente de los sistemas dentro del hogar, permite a los usuarios controlar y monitorear dispositivos desde cualquier lugar mediante aplicaciones móviles o comandos de voz. Por otro lado, la inmótica extiende estos conceptos a edificios comerciales e industriales, optimizando la gestión de recursos y servicios para mejorar la eficiencia energética, la seguridad y el confort.

Los ambientes inteligentes y las ciudades inteligentes representan la culminación de estos avances, integrando una amplia gama de sistemas embebidos para crear entornos urbanos altamente eficientes y sostenibles. Desde la gestión inteligente de la energía y los residuos hasta el transporte público autónomo y la seguridad ciudadana basada en datos, estas iniciativas están redefiniendo la forma en que vivimos, trabajamos y nos relacionamos con nuestras comunidades.

A pesar de las enormes oportunidades que ofrece la integración de sistemas embebidos en entornos inteligentes, también existen desafíos significativos que deben abordarse. La interoperabilidad de dispositivos, la seguridad cibernética, la privacidad de datos y la brecha digital son solo algunas de las preocupaciones que requieren atención. Además, el rápido ritmo de innovación tecnológica demanda una inversión continua en investigación, desarrollo y capacitación para garantizar que los sistemas embebidos sigan siendo relevantes y efectivos en un entorno en constante evolución.

En conclusión, la integración de sistemas embebidos en entornos inteligentes representa una evolución significativa en el panorama tecnológico global. Al aprovechar la automatización, la inteligencia y la conectividad, estos sistemas están transformando la forma en que interactuamos con nuestro entorno, creando nuevas oportunidades para la eficiencia, la sostenibilidad y la calidad de vida. Sin embargo, para maximizar su potencial, es imperativo abordar los desafíos y trabajar en colaboración para construir un futuro inteligente y conectado para todos.

**Fuente de información:**

[**https://es.wikipedia.org/wiki/Electr%C3%B3nica**](https://es.wikipedia.org/wiki/Electr%C3%B3nica)

[**https://www.khanacademy.org/science/chemistry/electronic-structure-of-atoms**](https://www.khanacademy.org/science/chemistry/electronic-structure-of-atoms)

[**https://www.tutorialspoint.com/basic\_electronics/index.htm**](https://www.tutorialspoint.com/basic_electronics/index.htm)

[**https://es.wikipedia.org/wiki/Sistema\_embebido**](https://es.wikipedia.org/wiki/Sistema_embebido)

[**https://www.freertos.org/**](https://www.freertos.org/)

[**https://www.arm.com/glossary/embedded-system-design**](https://www.arm.com/glossary/embedded-system-design)

[**https://es.wikipedia.org/wiki/Automatizaci%C3%B3n**](https://es.wikipedia.org/wiki/Automatizaci%C3%B3n)

[**https://towardsdatascience.com/but-what-exactly-is-ai-59454770d39b**](https://towardsdatascience.com/but-what-exactly-is-ai-59454770d39b)

[**https://ifr.org/**](https://ifr.org/)

[**https://es.wikipedia.org/wiki/Dom%C3%B3tica**](https://es.wikipedia.org/wiki/Dom%C3%B3tica)

[**https://asean.org/our-communities/asean-smart-cities-network/**](https://asean.org/our-communities/asean-smart-cities-network/)

[**https://www.itu.int/dms\_pub/itu-t/oth/0b/04/T0B0400004F2C01PDFE.pdf**](https://www.itu.int/dms_pub/itu-t/oth/0b/04/T0B0400004F2C01PDFE.pdf)

[**https://www.smartcitiescouncil.com/**](https://www.smartcitiescouncil.com/)

[**https://www.highintegritysystems.com/rtos/what-is-an-rtos/**](https://www.highintegritysystems.com/rtos/what-is-an-rtos/)

[**https://embeddedhardwaredesign.com/how-to-design-hardware-for-embedded-systems/**](https://embeddedhardwaredesign.com/how-to-design-hardware-for-embedded-systems/)

# 3. Programas

**1- Tipos de datos**

**Descripción**:

Este programa de Arduino tiene como objetivo mostrar el uso de diferentes tipos de datos en un entorno de microcontroladores. Los tipos de datos incluyen enteros, decimales, booleanos y caracteres. Se configuran variables de cada tipo, se les asignan valores y se imprimen estos valores en el monitor serie.

**Introducción**:

En esta práctica, se abordan los fundamentos teóricos de los tipos de datos en Arduino y su utilidad en la programación de microcontroladores. Estos conceptos son esenciales para manejar la memoria y optimizar el rendimiento del programa.

**Fundamentos Teóricos:**

1. **Enteros (int, long, short, byte):** Son tipos de datos numéricos sin decimales. Se utilizan para contar, medir o indexar.
2. **Decimales (double, float):** Permiten almacenar números con punto decimal, útiles en cálculos que requieren precisión.
3. **Booleano (bool):** Representa valores de verdad, útil para decisiones lógicas.
4. **Caracteres (String, char):** Almacenan caracteres individuales o cadenas de texto.

* Incluir Instrucciones, Componentes o Procedimientos utilizados:
  + Instrucción 1: Declaración de Variables
    - Objetivo: Definir y asignar valores a diferentes tipos de datos.
    - Sintaxis: int nombreVariable = valor;
    - Ejemplo: int entero = 10;
  + Instrucción 2: Configuración con el puerto serial
    - Objetico: configurar la velocidad de transmisión de datos a través del puerto serial.
    - Sintaxis: Serial.begin(velocidad);
    - Ejemplo: Serial.begin(9600);
  + Instrucción 3: Impresión en el monitor serial
    - Objetivo: Enviar datos al monitor serial para su visualización
    - Sintaxis: Serial.println(valor);
    - Ejemplo: Serial.println(“texto a imprimir”);
* Incluir información como:

Nos servirá para conocer los tipos de datos que son compatibles con Arduino.

**Desarrollo:**

**Paso 1: Declaración de Variables**

Se declaran variables de diferentes tipos de datos y se les asignan valores iniciales.

**Paso 2: Configuración de la Velocidad de Transmisión Serie**

En el **setup()**, se establece la velocidad de transmisión en 9600 baudios para la comunicación serie.

**Paso 3: Impresión de Valores en el Monitor Serie**

Dentro del **setup()**, se imprimen los valores de las variables utilizando la función **Serial.println()**, que envía los datos al monitor serie.

**Paso 4: Loop Vacío**

El **loop()** se deja vacío ya que no se necesita repetir ninguna acción continuamente.

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamenteDemostración: Imagen de la pantalla de un ordenador portátil

Descripción generada automáticamente con confianza media**

**2- Encendido y apagado de un Led**

**Descripción:**

Este programa de Arduino tiene como objetivo encender y apagar un LED conectado al pin digital 12 de la placa Arduino. El LED se enciende y se apaga con un intervalo de 1 segundo, utilizando las funciones **digitalWrite** y **delay**.

**Introducción:**

En esta práctica, se abordan los fundamentos teóricos del control de actuadores, específicamente LEDs, utilizando la plataforma Arduino. Estos conceptos son esenciales para comprender cómo interactuar con componentes electrónicos básicos.

**Fundamentos Teóricos:**

1. **Pin Digital:** Puntos de conexión en el Arduino que pueden ser configurados como entradas o salidas.
2. **LED (Light Emitting Diode):** Un diodo que emite luz cuando pasa corriente a través de él.
3. **Funciones Básicas:**
   * **pinMode(pin, mode):** Configura el pin especificado para actuar como entrada (**INPUT**) o salida (**OUTPUT**).
   * **digitalWrite(pin, value):** Escribe un valor alto (**HIGH**) o bajo (**LOW**) en un pin digital.
   * **delay(ms):** Detiene la ejecución del programa durante un número específico de milisegundos.

Instrucciones, Componentes y Procedimientos Utilizados:

Instrucción 1: Declaración de Variables

* Objetivo: Definir el pin al que está conectado el LED.
* Sintaxis: int nombreVariable = Valor;
* Ejemplo: int led = 12;

Instrucción 2: Configuración del Modo del Pin

* Objetivo: Configurar el pin como salida para controlar el LED.
* Sintaxis: pinMode(pin, mode);
* Ejemplo: pinMode(led, OUTPUT);

Instrucción 3: Escribir en el Pin Digital

* Objetivo: Encender o apagar el LED.
* Sintaxis: digitalWrite(pin, value);
* Ejemplo: digitalWrite(led, HIGH);
* DigitalWrite(led, Low);

Instrucción 3: Escribir en el Pin Digital

* Objetivo: Encender o apagar el LED.
* Sintaxis: delay(ms);
* Ejemplo: delay(1000);

**Desarrollo:**

**Paso 1: Declaración de Variables**

Se declara una variable para el pin al que está conectado el LED.

**Paso 2: Configuración del Modo del Pin**

En el setup(), se configura el pin 12 como salida utilizando la función pinMode.

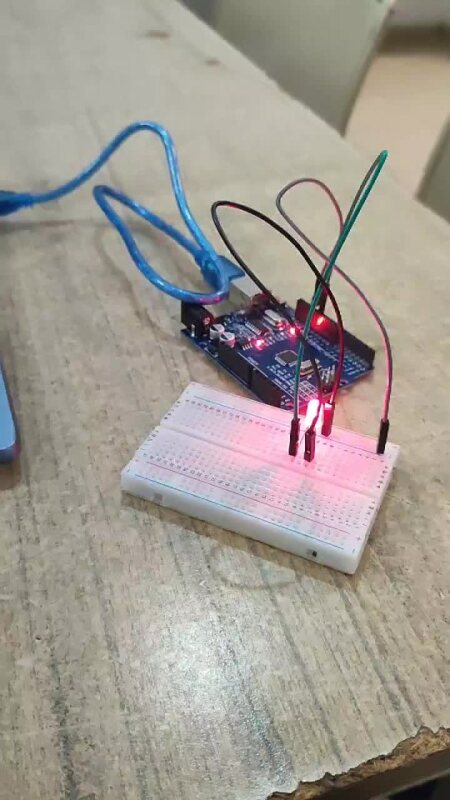
**Paso 3: Ciclo Principal (Loop)**

En el loop(), se enciende el LED estableciendo el pin 12 en alto (HIGH), se espera 1 segundo (delay(1000)), y luego se apaga el LED estableciendo el pin 12 en bajo (LOW), seguido por otra espera de 1 segundo.

**Demostración:**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**[](https://alumnosuatedu-my.sharepoint.com/:v:/g/personal/a2163226011_alumnos_uat_edu_mx/EXL-QIdHmvpFl9YYco6VPh8B5CQlsPnmcN4SoXpGc3ZtfA?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=8zLwdB)**

**3-Impresión de hola mundo**

**Descripción:**

Este programa de Arduino tiene como objetivo enviar un mensaje de "Hola mundo" al monitor serie de la computadora cada 500 milisegundos. Utiliza la comunicación serie UART para transmitir datos entre la placa Arduino y la computadora.

**Introducción:**

En esta práctica, se abordan los fundamentos teóricos de la comunicación serie utilizando la plataforma Arduino. Estos conceptos son esenciales para la transmisión y recepción de datos entre el microcontrolador y otros dispositivos.

**Fundamentos Teóricos:**

1. **Comunicación Serie (UART):** Un protocolo de comunicación que permite la transmisión de datos bit a bit. Es comúnmente utilizado para la comunicación entre la placa Arduino y la computadora.
2. **Monitor Serie:** Una herramienta que permite ver los datos enviados desde la placa Arduino a la computadora.
3. **Funciones Básicas:**
   * **Serial.begin(velocidad):** Inicializa la comunicación serie a una velocidad especificada en baudios.
   * **Serial.println(mensaje):** Envía un mensaje al monitor serie seguido de un salto de línea.
   * **delay(ms):** Detiene la ejecución del programa durante un número específico de milisegundos.

**Instrucciones, Componentes y Procedimientos Utilizados:**

Instrucción 1: Inicialización de la Comunicación Serie

* Objetivo: Configurar la velocidad de transmisión de datos en baudios.
* Sintaxis: Serial.begin(velocidad);
* Ejemplo: Serial.begin(9600);

Instrucción 2: Envío de Datos al Monitor Serie

* Objetivo: Enviar un mensaje de texto al monitor serie.
* Sintaxis: Serial.println(mensaje);
* Ejemplo: Serial.println(“Hola mundo”);

Instrucción 3: Demora en la Ejecución

Objetivo: Crear una pausa entre los envíos de mensajes.

Sintaxis: Delay(ms);

Ejemplo: delay(500);

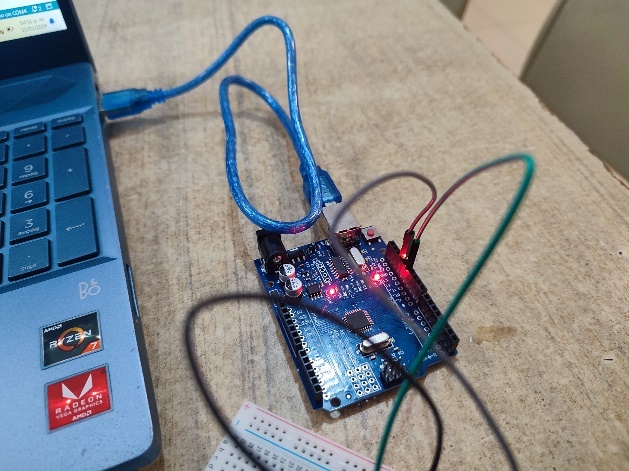
**Desarrollo:**

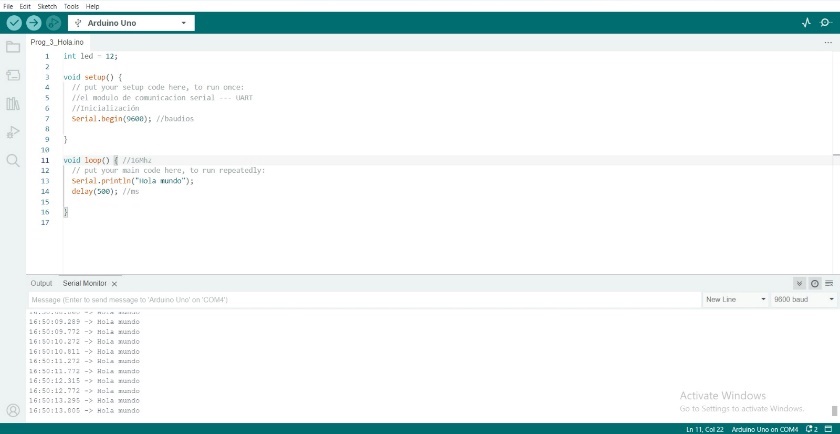
**Paso 1: Inicialización de la Comunicación Serie**

En el **setup()**, se inicializa la comunicación serie a 9600 baudios utilizando la función **Serial.begin**.

**Paso 2: Ciclo Principal (Loop)**

En el **loop()**, se envía el mensaje "Hola mundo" al monitor serie con la función **Serial.println** y se espera 500 milisegundos antes de repetir el proceso.

**Pantalla de computadora con teclado

Descripción generada automáticamenteDemostración:**

**4- Encendido y Apagado Led BlinkLed-serial**

**Descripción:**

Este programa de Arduino tiene como objetivo encender y apagar un LED conectado al pin digital 12 de la placa Arduino y enviar mensajes al monitor serie para indicar el estado del LED. El LED se enciende y se apaga con un intervalo de 1 segundo, y los mensajes correspondientes se imprimen en el monitor serie.

**Introducción:**

En esta práctica, se abordan los fundamentos teóricos del control de actuadores, específicamente LEDs, y la comunicación serie utilizando la plataforma Arduino. Estos conceptos son esenciales para la interacción con componentes electrónicos y la transmisión de datos entre el microcontrolador y otros dispositivos.

**Fundamentos Teóricos:**

1. **Pin Digital:** Puntos de conexión en el Arduino que pueden ser configurados como entradas o salidas.
2. **LED (Light Emitting Diode):** Un diodo que emite luz cuando pasa corriente a través de él.
3. **Comunicación Serie (UART):** Un protocolo de comunicación que permite la transmisión de datos bit a bit.
4. **Monitor Serie:** Una herramienta que permite ver los datos enviados desde la placa Arduino a la computadora.
5. **Funciones Básicas:**
   * **pinMode(pin, mode):** Configura el pin especificado para actuar como entrada (**INPUT**) o salida (**OUTPUT**).
   * **digitalWrite(pin, value):** Escribe un valor alto (**HIGH**) o bajo (**LOW**) en un pin digital.
   * **Serial.begin(velocidad):** Inicializa la comunicación serie a una velocidad especificada en baudios.
   * **Serial.println(mensaje):** Envía un mensaje al monitor serie seguido de un salto de línea.
   * **delay(ms):** Detiene la ejecución del programa durante un número específico de milisegundos.

**Instrucciones, Componentes y Procedimientos Utilizados:**

Instrucción 1: Declaración de Variables

* Objetivo: Definir el pin al que está conectado el LED.
* Sintaxis: Int nombreVariable = valor;
* Ejemplo: int led = 12;

Instrucción 2: Configuración del Modo del Pin

* Objetivo: Configurar el pin como salida para controlar el LED.
* Sintaxis: pinMode(pin, mode);
* Ejemplo: pinMode(led, output);

Instrucción 3: Inicialización de la Comunicación Serie

* Objetivo: Configurar la velocidad de transmisión de datos en baudios.
* Sintaxis: Serial.begin (velocidad);
* Ejemplo: Serial.begin(9600);

Instrucción 4: Escribir en el Pin Digital

* Objetivo: Encender o apagar el LED.
* Sintaxis: digitalWrite(pin, value);
* Ejemplo: digitalWrite(led, HIGH); // Encender el LED
* digitalWrite(led, LOW); // Apagar el LED

Instrucción 5: Envío de Datos al Monitor Serie

* Objetivo: Enviar un mensaje de texto al monitor serie.
* Sintaxis: serial.println(mensaje);
* Ejemplo: Serial.println(“Led Encendido”);

Instrucción 6: Demora en la Ejecución

* Objetivo: Crear una pausa entre las acciones de encender y apagar el LED.
* Sintaxis: delay(ms);
* Ejemplo: delay(1000);

**Desarrollo:**

**Paso 1:** Declaración de Variables

Se declara una variable para el pin al que está conectado el LED.

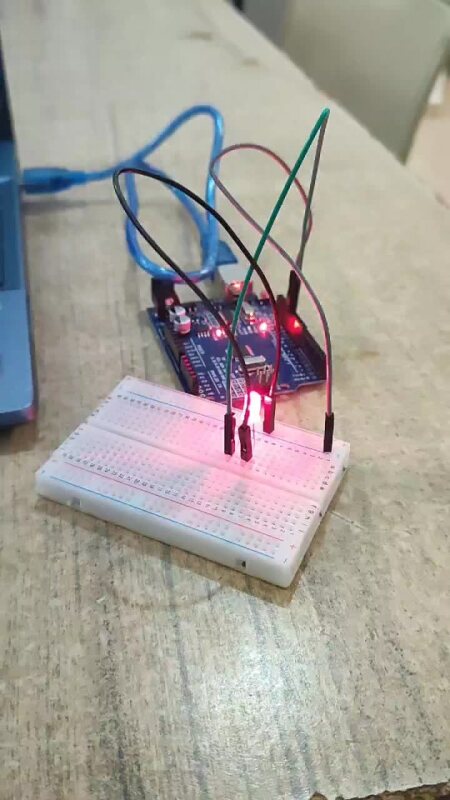
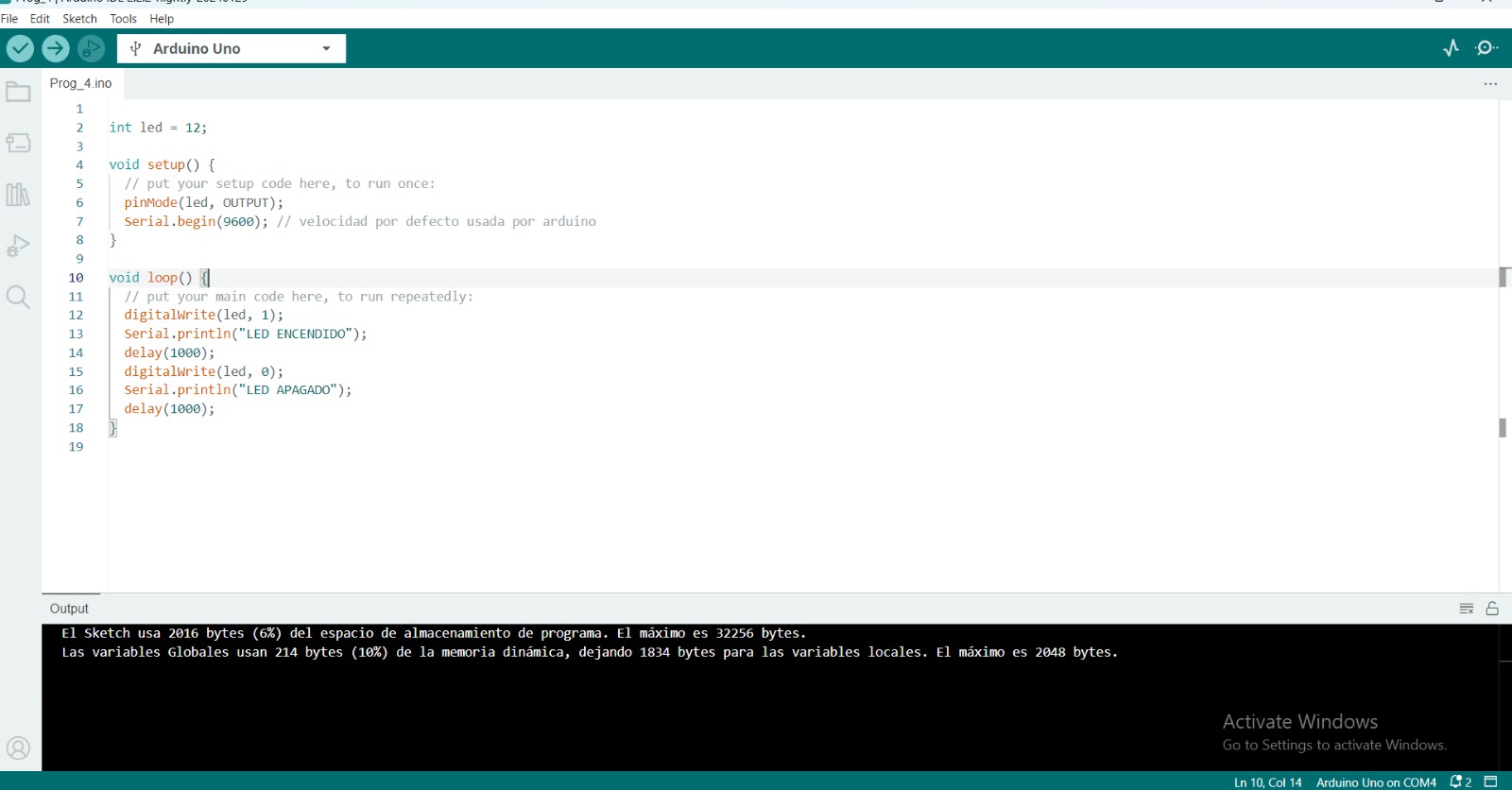
**Paso 2:** Configuración del Modo del Pin y la Comunicación Serie

En el setup(), se configura el pin 12 como salida utilizando la función pinMode y se inicializa la comunicación serie a 9600 baudios utilizando la función Serial.begin.

**Paso 3:** Ciclo Principal (Loop)

En el loop(), se enciende el LED estableciendo el pin 12 en alto (HIGH), se envía el mensaje "LED ENCENDIDO" al monitor serie con la función Serial.println, se espera 1 segundo (delay(1000)), y luego se apaga el LED estableciendo el pin 12 en bajo (LOW), seguido por el mensaje "LED APAGADO" y otra espera de 1 segundo.

**Demostración:**

**[](https://alumnosuatedu-my.sharepoint.com/:v:/g/personal/a2163226011_alumnos_uat_edu_mx/EW0NIxI7IMhBs0ti8Za0XVoBtG2ndqdCH_NNtjI7taXX3A?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=R3qiSW)**

**5-Encendido y apagado led buffer**

**Descripción:**

Este programa de Arduino permite encender y apagar un LED conectado al pin digital 12 mediante comandos enviados desde el monitor serie. Los comandos "1" y "0" se utilizan para encender y apagar el LED, respectivamente.

**Introducción:**

En esta práctica, se abordan los fundamentos teóricos del control de actuadores y la interacción mediante la comunicación serie utilizando la plataforma Arduino. Estos conceptos son esenciales para la creación de interfaces de control simples entre el microcontrolador y el usuario.

**Fundamentos Teóricos:**

1. **Pin Digital:** Puntos de conexión en el Arduino que pueden ser configurados como entradas o salidas.
2. **LED (Light Emitting Diode):** Un diodo que emite luz cuando pasa corriente a través de él.
3. **Comunicación Serie (UART):** Un protocolo de comunicación que permite la transmisión de datos bit a bit.
4. **Monitor Serie:** Una herramienta que permite ver los datos enviados desde la placa Arduino a la computadora y enviar datos a la placa.
5. **Funciones Básicas:**
   * **pinMode(pin, mode):** Configura el pin especificado para actuar como entrada (**INPUT**) o salida (**OUTPUT**).
   * **digitalWrite(pin, value):** Escribe un valor alto (**HIGH**) o bajo (**LOW**) en un pin digital.
   * **Serial.begin(velocidad):** Inicializa la comunicación serie a una velocidad especificada en baudios.
   * **Serial.println(mensaje):** Envía un mensaje al monitor serie seguido de un salto de línea.
   * **Serial.available():** Retorna el número de bytes disponibles para leer desde el buffer serie.
   * **Serial.readString():** Lee una cadena de caracteres desde el buffer serie.
   * **delay(ms):** Detiene la ejecución del programa durante un número específico de milisegundos.

**Instrucciones, Componentes y Procedimientos Utilizados:**

**Instrucción 1: Declaración de Variables**

* **Objetivo:** Definir el pin al que está conectado el LED.
* **Sintaxis:**int nombreVariable = valor;
* **Ejemplo:**int led = 12;

**Instrucción 2: Configuración del Modo del Pin y la Comunicación Serie**

* **Objetivo:** Configurar el pin como salida para controlar el LED y configurar la comunicación serie.
* **Sintaxis:**pinMode(pin, mode); Serial.begin(velocidad);
* **Ejemplo:**pinMode(led, OUTPUT); Serial.begin(9600);

**Instrucción 3: Escribir en el Pin Digital**

* **Objetivo:** Encender o apagar el LED según el comando recibido.
* **Sintaxis:**digitalWrite(pin, value);
* **Ejemplo:**digitalWrite(led, HIGH); // Encender el LED digitalWrite(led, LOW); // Apagar el LED

**Instrucción 4: Envío de Datos al Monitor Serie**

* **Objetivo:** Enviar mensajes de texto al monitor serie.
* **Sintaxis:**Serial.println(mensaje);
* **Ejemplo:**Serial.println("LED ENCENDIDO");

**Instrucción 5: Leer Datos desde el Monitor Serie**

* **Objetivo:** Leer y procesar comandos enviados desde el monitor serie.
* **Sintaxis:**if (Serial.available() > 0) { int resultado = Serial.readString().toInt(); }
* **Ejemplo:**if (Serial.available() > 0) { int resultado = Serial.readString().toInt(); }

**Instrucción 6: Demora en la Ejecución**

* **Objetivo:** Crear una pequeña pausa para permitir la lectura y procesamiento de los comandos.
* **Sintaxis:**delay(ms);
* **Ejemplo:**delay(10); // Pausa de 10 milisegundos

**Desarrollo:**

**Paso 1: Declaración de Variables**

Se declara una variable para el pin al que está conectado el LED.

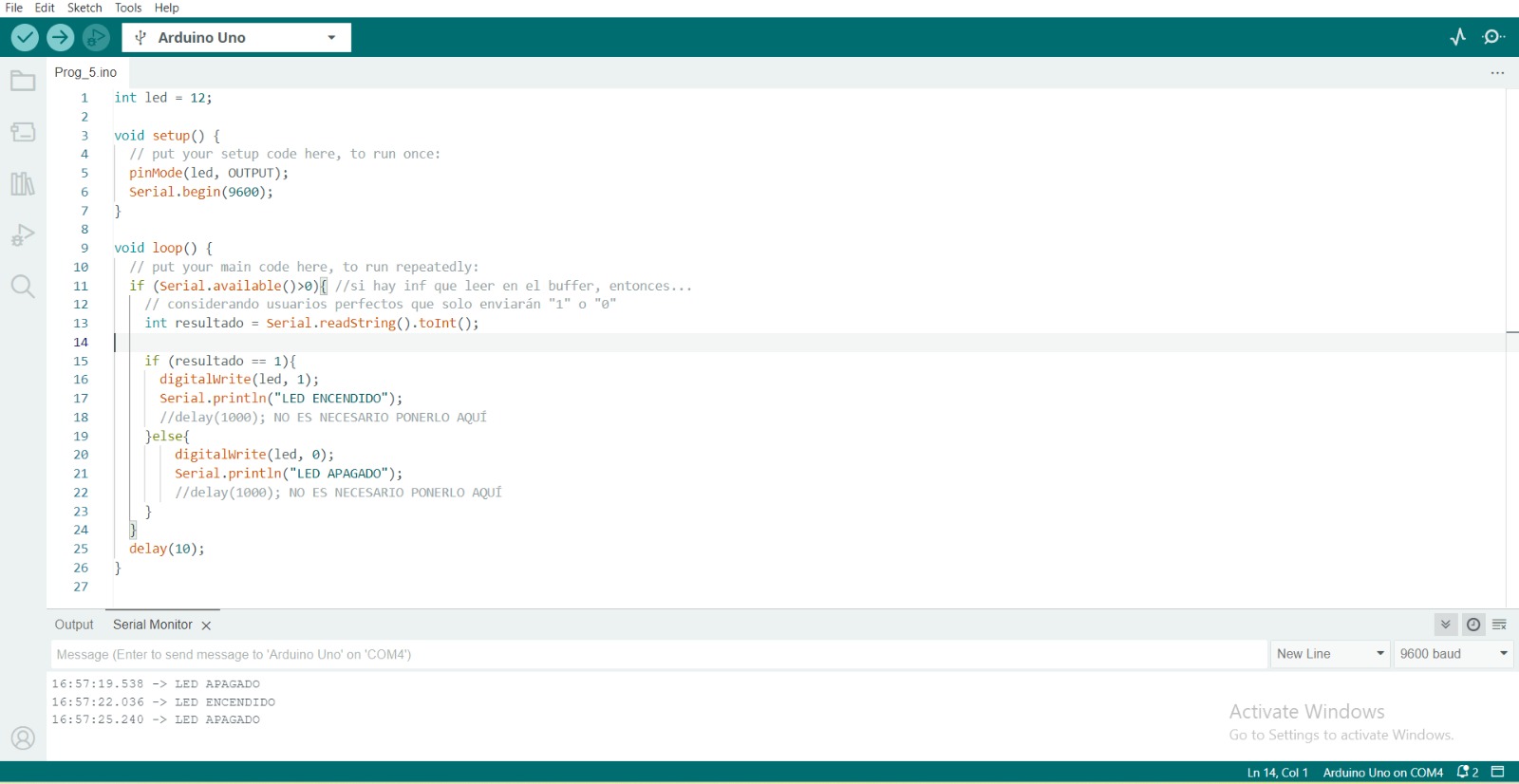
**Paso 2: Configuración del Modo del Pin y la Comunicación Serie**

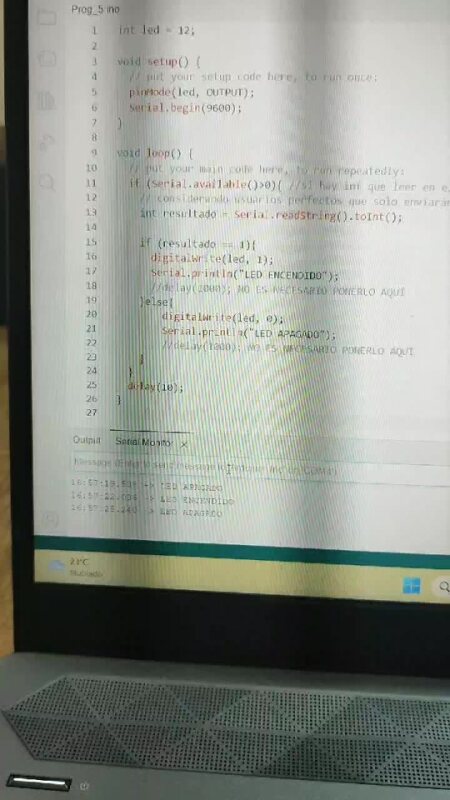
En el **setup()**, se configura el pin 12 como salida utilizando la función **pinMode** y se inicializa la comunicación serie a 9600 baudios utilizando la función **Serial.begin**.

**Paso 3: Ciclo Principal (Loop)**

En el **loop()**, se envía el mensaje "Para encender el led presione 1 y para apagarlo presione 0" al monitor serie con la función **Serial.println**. Luego, se verifica si hay datos disponibles en el buffer serie con **Serial.available**. Si hay datos disponibles, se leen y se convierten a un entero utilizando **Serial.readString().toInt**. Dependiendo del valor leído, se enciende o apaga el LED y se envía un mensaje correspondiente al monitor serie. Finalmente, se espera 10 milisegundos antes de repetir el proceso.

**Demostración:**

****

**[](https://alumnosuatedu-my.sharepoint.com/:v:/g/personal/a2163226011_alumnos_uat_edu_mx/EeDb7yubpeBKgsBNNcHi2LABeOrgJ2U5yEgTXnnhifBDHw?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=V3eetS)**

**6- Encendido de leds en ciclo**

**Descripción:**

Este programa de Arduino permite controlar tres LEDs conectados a los pines digitales 9, 10 y 11 mediante comandos enviados desde el monitor serie. Los comandos "0", "1" y "2" se utilizan para encender los LEDs correspondientes, mientras que cualquier otro comando enviará un mensaje de error al monitor serie.

**Introducción:**

En esta práctica, se abordan los fundamentos teóricos del control de múltiples actuadores y la interacción mediante la comunicación serie utilizando la plataforma Arduino. Estos conceptos son esenciales para la creación de interfaces de control avanzadas entre el microcontrolador y el usuario.

**Fundamentos Teóricos:**

1. **Pin Digital:** Puntos de conexión en el Arduino que pueden ser configurados como entradas o salidas.
2. **LED (Light Emitting Diode):** Un diodo que emite luz cuando pasa corriente a través de él.
3. **Comunicación Serie (UART):** Un protocolo de comunicación que permite la transmisión de datos bit a bit.
4. **Monitor Serie:** Una herramienta que permite ver los datos enviados desde la placa Arduino a la computadora y enviar datos a la placa.
5. **Funciones Básicas:**
   * **pinMode(pin, mode):** Configura el pin especificado para actuar como entrada (**INPUT**) o salida (**OUTPUT**).
   * **digitalWrite(pin, value):** Escribe un valor alto (**HIGH**) o bajo (**LOW**) en un pin digital.
   * **Serial.begin(velocidad):** Inicializa la comunicación serie a una velocidad especificada en baudios.
   * **Serial.setTimeout(tiempo):** Configura el tiempo de espera para la comunicación serie.
   * **Serial.println(mensaje):** Envía un mensaje al monitor serie seguido de un salto de línea.
   * **Serial.available():** Retorna el número de bytes disponibles para leer desde el buffer serie.
   * **Serial.readString():** Lee una cadena de caracteres desde el buffer serie.
   * **delay(ms):** Detiene la ejecución del programa durante un número específico de milisegundos.

**Instrucciones, Componentes y Procedimientos Utilizados:**

**Instrucción 1: Declaración de Variables**

* **Objetivo:** Definir los pines a los que están conectados los LEDs.
* **Sintaxis:**int nombreVariable = valor;
* **Ejemplo:**int led1 = 9; int led2 = 10; int led3 = 11;

**Instrucción 2: Configuración del Modo del Pin y la Comunicación Serie**

* **Objetivo:** Configurar los pines como salidas para controlar los LEDs y configurar la comunicación serie.
* **Sintaxis:**pinMode(pin, mode); Serial.begin(velocidad); Serial.setTimeout(tiempo);
* **Ejemplo:**pinMode(led1, OUTPUT); pinMode(led2, OUTPUT); pinMode(led3, OUTPUT); Serial.begin(9600); Serial.setTimeout(10);

**Instrucción 3: Escribir en los Pines Digitales**

* **Objetivo:** Encender o apagar los LEDs según el comando recibido.
* **Sintaxis:**digitalWrite(pin, value);
* **Ejemplo:**digitalWrite(led1, HIGH); // Encender el LED 1 digitalWrite(led1, LOW); // Apagar el LED 1

**Instrucción 4: Envío de Datos al Monitor Serie**

* **Objetivo:** Enviar mensajes de texto al monitor serie.
* **Sintaxis:**Serial.println(mensaje);
* **Ejemplo:**Serial.println("error");

**Instrucción 5: Leer Datos desde el Monitor Serie**

* **Objetivo:** Leer y procesar comandos enviados desde el monitor serie.
* **Sintaxis:**if (Serial.available() > 0) { int v = Serial.readString().toInt(); }
* **Ejemplo:**if (Serial.available() > 0) { int v = Serial.readString().toInt(); }

**Desarrollo:**

**Paso 1:** Declaración de Variables

Se declaran variables para los pines a los que están conectados los LEDs.

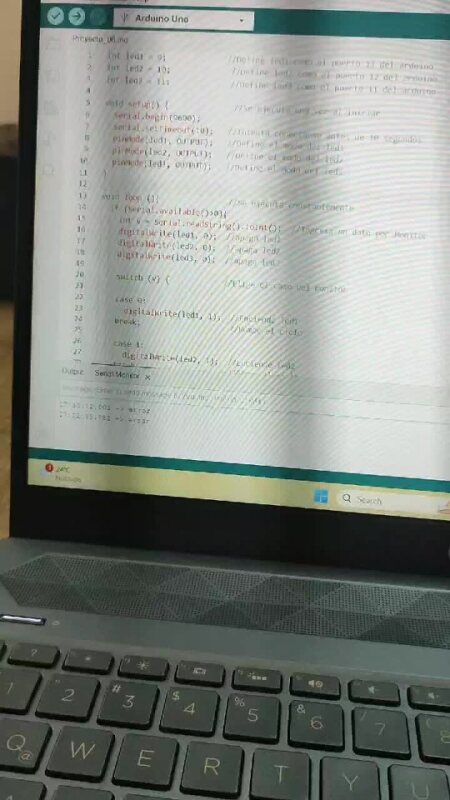
**Paso 2:** Configuración del Modo del Pin y la Comunicación Serie

En el setup(), se configuran los pines 9, 10 y 11 como salidas utilizando la función pinMode y se inicializa la comunicación serie a 9600 baudios utilizando la función Serial.begin. También se configura un tiempo de espera para la comunicación serie utilizando la función Serial.setTimeout.

**Paso 3:** Ciclo Principal (Loop)

En el loop(), se verifica si hay datos disponibles en el buffer serie con Serial.available. Si hay datos disponibles, se leen y se convierten a un entero utilizando Serial.readString().toInt. Luego, se apagan todos los LEDs utilizando la función digitalWrite. Dependiendo del valor leído, se enciende el LED correspondiente utilizando una estructura switch. Si el valor no corresponde a "0", "1" o "2", se envía un mensaje de error al monitor serie utilizando la función Serial.println.

**Demostración:**

**[](https://alumnosuatedu-my.sharepoint.com/:v:/g/personal/a2163226011_alumnos_uat_edu_mx/EezkxFS2X3lMsSQMbZ7nSY4BqX5GXW-UKjlFfDiyWOrzbw?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=BfdyAU)**

**7- Encendido y apagado de leds por iteración**

**Descripción:**

Este programa de Arduino controla cuatro LEDs conectados a los pines digitales 9, 10, 11 y 12, encendiéndolos y apagándolos secuencialmente con un intervalo de un segundo.

**Introducción:**

En esta práctica, se utilizan conceptos fundamentales de control de actuadores, bucles y temporización en la plataforma Arduino. Estos conceptos permiten crear efectos de iluminación secuenciales utilizando múltiples LEDs.

**Fundamentos Teóricos:**

1. **Pin Digital:** Puntos de conexión en el Arduino que pueden ser configurados como entradas o salidas.
2. **LED (Light Emitting Diode):** Un diodo que emite luz cuando pasa corriente a través de él.
3. **Bucle for:** Estructura de control que permite ejecutar un bloque de código un número específico de veces.
4. **Funciones Básicas:**
   * **pinMode(pin, mode):** Configura el pin especificado para actuar como entrada (**INPUT**) o salida (**OUTPUT**).
   * **digitalWrite(pin, value):** Escribe un valor alto (**HIGH**) o bajo (**LOW**) en un pin digital.
   * **delay(ms):** Detiene la ejecución del programa durante un número específico de milisegundos.

**Instrucciones, Componentes y Procedimientos Utilizados:**

**Instrucción 1: Declaración de Variables**

* **Objetivo:** Definir los pines a los que están conectados los LEDs utilizando un array.
* **Sintaxis:**int nombreArray[tamaño] = {valores};
* **Ejemplo:**int leds[4] = {9, 10, 11, 12};

**Instrucción 2: Configuración del Modo del Pin**

* **Objetivo:** Configurar los pines como salidas para controlar los LEDs.
* **Sintaxis:**pinMode(pin, mode);
* **Ejemplo:**pinMode(leds[i], OUTPUT);

**Instrucción 3: Escribir en los Pines Digitales**

* **Objetivo:** Encender y apagar los LEDs secuencialmente.
* **Sintaxis:**digitalWrite(pin, value);
* **Ejemplo:**digitalWrite(leds[i], HIGH); // Encender el LED digitalWrite(leds[i], LOW); // Apagar el LED

**Instrucción 4: Demora en la Ejecución**

* **Objetivo:** Crear un intervalo de tiempo entre la activación de cada LED.
* **Sintaxis:**delay(ms);
* **Ejemplo:**delay(1000); // Pausa de 1000 milisegundos

**Desarrollo:**

**Paso 1:** Declaración de Variables

Se declara un array llamado leds que contiene los pines a los que están conectados los LEDs.

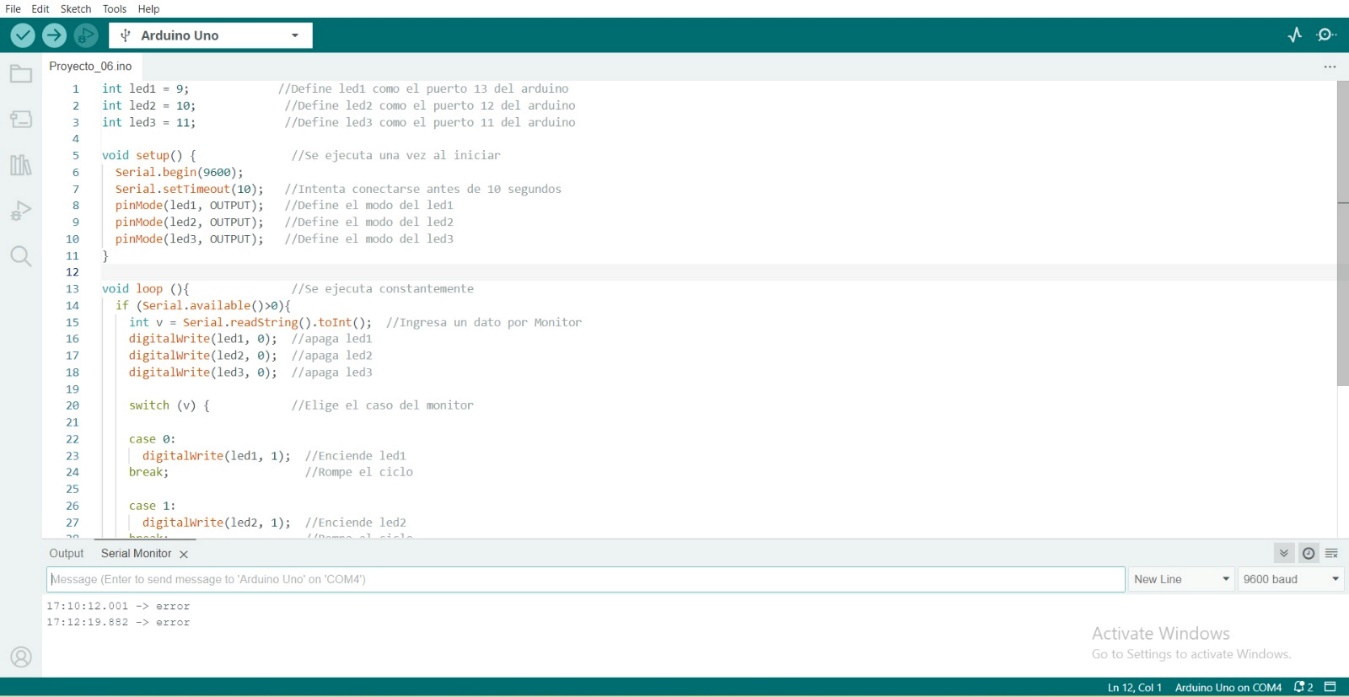
**Paso 2:** Configuración del Modo del Pin

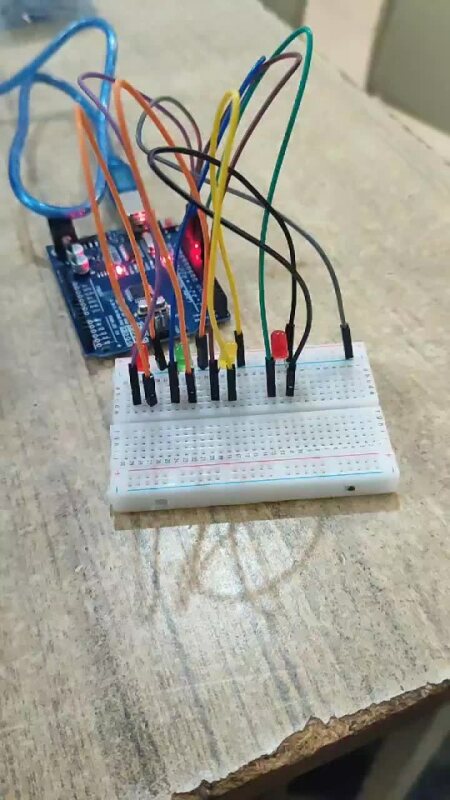
En el setup(), se utiliza un bucle for para configurar los pines 9, 10, 11 y 12 como salidas utilizando la función pinMode.

**Paso 3:** Ciclo Principal (Loop)

En el loop(), se utiliza otro bucle for para encender y apagar los LEDs secuencialmente. Cada LED se enciende utilizando la función digitalWrite y luego se apaga después de un segundo. Este proceso se repite continuamente.

**Demostración:**

****

**[](https://alumnosuatedu-my.sharepoint.com/:v:/g/personal/a2163226011_alumnos_uat_edu_mx/Ec1boxhjUsFHnv90nd59li0B636tLJdEKfGlevRr4E9x5Q?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=SRUxTU)**

**8- Encendido y apagado leds por cadena**

**Descripción:**

Este programa de Arduino permite controlar cuatro LEDs conectados a los pines digitales 9, 10, 11 y 12 mediante comandos enviados desde el monitor serie. Se envía un número del 0 al 3 para encender el LED correspondiente mientras se apagan los otros LEDs.

**Introducción:**

En esta práctica se utiliza la comunicación serie para controlar múltiples LEDs en una placa Arduino. Se hace uso de arrays, bucles y funciones básicas de entrada/salida para manejar los actuadores (LEDs).

**Fundamentos Teóricos:**

1. **Pin Digital:** Puntos de conexión en el Arduino que pueden ser configurados como entradas o salidas.
2. **LED (Light Emitting Diode):** Un diodo que emite luz cuando pasa corriente a través de él.
3. **Comunicación Serie (UART):** Un protocolo de comunicación que permite la transmisión de datos bit a bit.
4. **Array:** Estructura de datos que almacena múltiples valores en una sola variable.
5. **Funciones Básicas:**
   * **pinMode(pin, mode):** Configura el pin especificado para actuar como entrada (**INPUT**) o salida (**OUTPUT**).
   * **digitalWrite(pin, value):** Escribe un valor alto (**HIGH**) o bajo (**LOW**) en un pin digital.
   * **Serial.begin(velocidad):** Inicializa la comunicación serie a una velocidad especificada en baudios.
   * **Serial.setTimeout(tiempo):** Configura el tiempo de espera para la comunicación serie.
   * **Serial.available():** Retorna el número de bytes disponibles para leer desde el buffer serie.
   * **Serial.readString():** Lee una cadena de caracteres desde el buffer serie y la retorna como una cadena de texto.
   * **delay(ms):** Detiene la ejecución del programa durante un número específico de milisegundos.

**Instrucciones, Componentes y Procedimientos Utilizados:**

**Instrucción 1: Declaración de Variables**

* **Objetivo:** Definir los pines a los que están conectados los LEDs utilizando un array.
* **Sintaxis:**int nombreArray[tamaño] = {valores};
* **Ejemplo:**int leds[4] = {9, 10, 11, 12};

**Instrucción 2: Configuración del Modo del Pin y la Comunicación Serie**

* **Objetivo:** Configurar los pines como salidas para controlar los LEDs y configurar la comunicación serie.
* **Sintaxis:**pinMode(pin, mode); Serial.begin(velocidad); Serial.setTimeout(tiempo);
* **Ejemplo:**for (int i = 0; i < 4; i++) { pinMode(leds[i], OUTPUT); } Serial.begin(9600); Serial.setTimeout(10);

**Instrucción 3: Escribir en los Pines Digitales**

* **Objetivo:** Encender y apagar los LEDs según el comando recibido.
* **Sintaxis:**digitalWrite(pin, value);
* **Ejemplo:**digitalWrite(leds[v], HIGH); // Encender el LED correspondiente digitalWrite(leds[i], LOW); // Apagar los LEDs

**Instrucción 4: Leer Datos desde el Monitor Serie**

* **Objetivo:** Leer y procesar comandos enviados desde el monitor serie.
* **Sintaxis:**if (Serial.available() > 0) { int v = Serial.readString().toInt(); }
* **Ejemplo:**if (Serial.available() > 0) { int v = Serial.readString().toInt(); }

**Desarrollo:**

**Paso 1:** Declaración de Variables

Se declara un array llamado leds que contiene los pines a los que están conectados los LEDs.

**Paso 2:** Configuración del Modo del Pin y la Comunicación Serie

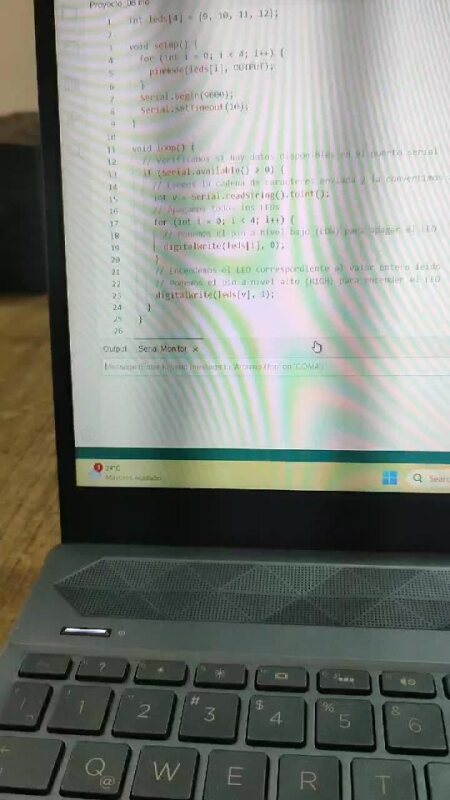
En el setup(), se utiliza un bucle for para configurar los pines 9, 10, 11 y 12 como salidas utilizando la función pinMode. Además, se inicializa la comunicación serie a 9600 baudios utilizando la función Serial.begin y se configura un tiempo de espera para la comunicación serie utilizando la función Serial.setTimeout.

**Paso 3:** Ciclo Principal (Loop)

En el loop(), se verifica si hay datos disponibles en el buffer serie con Serial.available. Si hay datos disponibles, se leen y se convierten a un entero utilizando Serial.readString().toInt. Luego, se apagan todos los LEDs utilizando un bucle for y la función digitalWrite. Dependiendo del valor leído, se enciende el LED correspondiente utilizando la función digitalWrite.

**Demostración:**

****

**[](https://alumnosuatedu-my.sharepoint.com/:v:/g/personal/a2163226011_alumnos_uat_edu_mx/EdWMIJNsVGZIkJceI9EX_KcBIQM6Hy7EV0sujlklFW8hMw?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=dfM0al)**

**9- Encendido y apagado led bucle**

**Descripción:**

Este programa de Arduino controla el brillo de un LED conectado al pin digital 10 utilizando PWM (Modulación por Ancho de Pulso). El LED aumenta y disminuye su brillo suavemente en un ciclo continuo.

**Introducción:**

En esta práctica se utilizan conceptos de modulación por ancho de pulso (PWM) para controlar la intensidad de un LED. La modulación PWM permite variar la cantidad de energía suministrada a un dispositivo, lo cual es útil para ajustar el brillo de un LED.

**Fundamentos Teóricos:**

1. **PWM (Pulse Width Modulation):** Una técnica para simular una señal analógica usando una señal digital que alterna entre encendido y apagado. La fracción de tiempo que la señal está encendida se denomina ciclo de trabajo.
2. **LED (Light Emitting Diode):** Un diodo que emite luz cuando pasa corriente a través de él.
3. **Funciones Básicas:**
   * **analogWrite(pin, value):** Escribe un valor analógico (PWM) en un pin. El valor puede ser entre 0 (apagado) y 255 (encendido completo).
   * **delay(ms):** Detiene la ejecución del programa durante un número específico de milisegundos.

**Instrucciones, Componentes y Procedimientos Utilizados:**

**Instrucción 1: Configuración del Modo del Pin**

* **Objetivo:** Configurar el pin al que está conectado el LED para utilizar PWM.
* **Sintaxis:**pinMode(pin, mode);
* **Ejemplo:**int led = 10;

**Instrucción 2: Escribir un Valor PWM**

* **Objetivo:** Ajustar el brillo del LED utilizando la función **analogWrite**.
* **Sintaxis:**analogWrite(pin, value);
* **Ejemplo:**analogWrite(led, i);

**Instrucción 3: Crear un Retardo**

* **Objetivo:** Crear un pequeño retraso para hacer la transición del brillo visible.
* **Sintaxis:**delay(ms);
* **Ejemplo:**delay(10);

**Desarrollo:**

**Paso 1:** Declaración de Variables

Se declara una variable led que contiene el pin al que está conectado el LED.

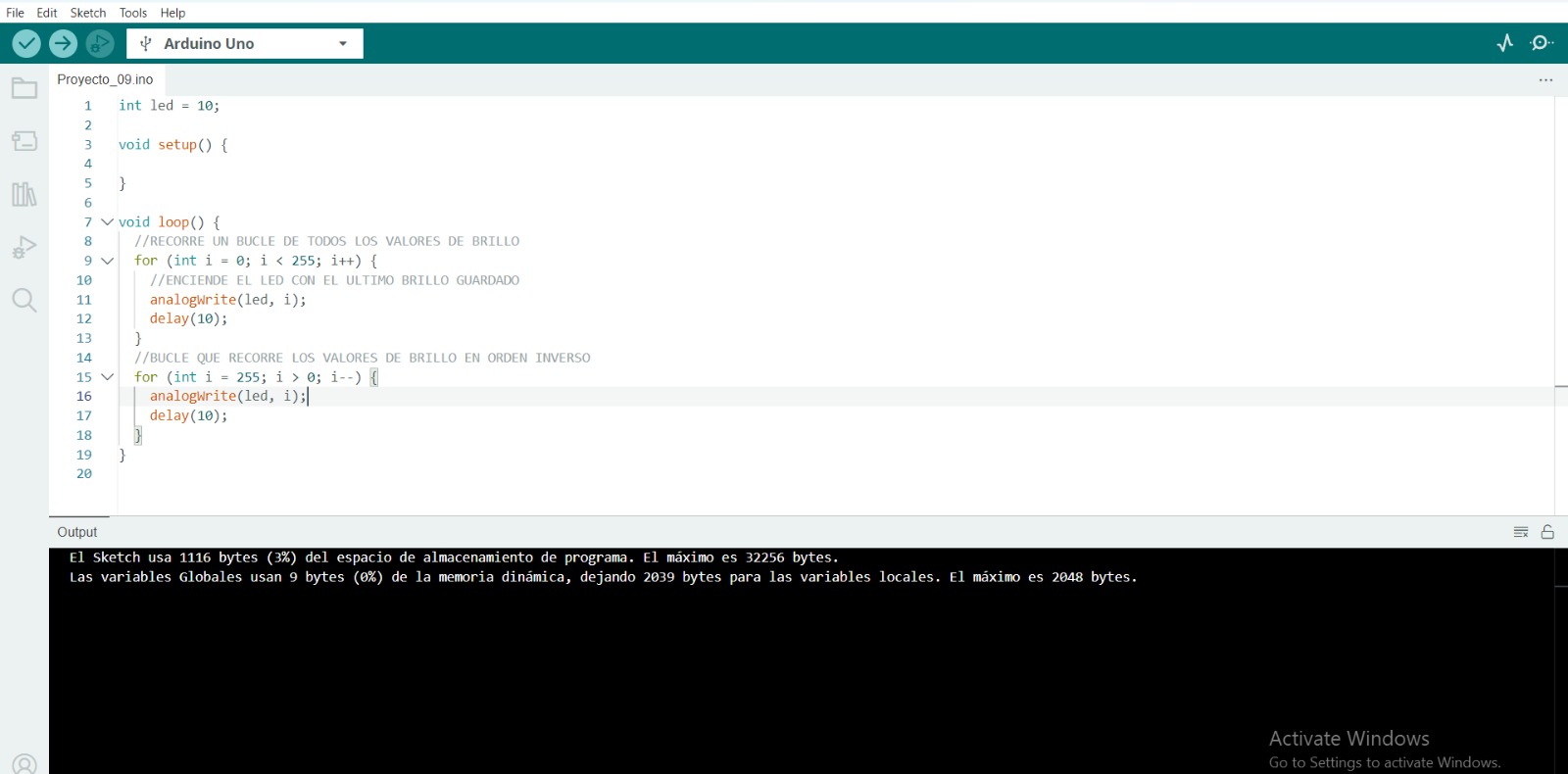
**Paso 2:** Configuración del Modo del Pin

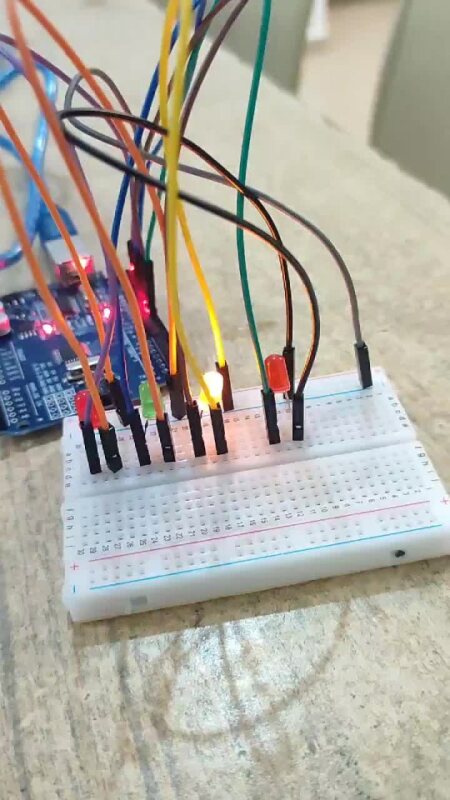
En el setup(), no es necesario configurar el pin como salida específicamente para PWM, ya que analogWrite se encarga de esto.

**Paso 3:** Ciclo Principal (Loop)

En el loop(), se utilizan dos bucles for para variar el brillo del LED. El primer bucle aumenta el valor de brillo de 0 a 255, y el segundo bucle lo disminuye de 255 a 0. Se utiliza analogWrite para ajustar el brillo del LED y delay para crear una transición suave.

**Demostración:**

****

**[](https://alumnosuatedu-my.sharepoint.com/:v:/g/personal/a2163226011_alumnos_uat_edu_mx/EUdHtmm0ltxFgAXDC21CsS4Bz-yiPyQUJOR8VXzci73AIg?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=gKPBgt)**

**10- Delay Arduino**

**Descripción:**

Este programa de Arduino muestra cómo obtener y mostrar el tiempo que ha transcurrido desde que se inició la placa en segundos. Utiliza la función **millis()** para obtener el tiempo en milisegundos y luego lo convierte en segundos para imprimirlo en el monitor serie cada segundo.

**Introducción:**

En esta práctica se utilizan funciones básicas de tiempo y comunicación serie para mostrar el tiempo transcurrido desde que se encendió la placa Arduino.

**Fundamentos Teóricos:**

1. **Función millis():** Devuelve el número de milisegundos que han pasado desde que la placa Arduino comenzó a ejecutar el programa actual.
2. **Comunicación Serie (UART):** Un protocolo de comunicación que permite la transmisión de datos bit a bit.
3. **Funciones Básicas:**
   * **Serial.begin(velocidad):** Inicializa la comunicación serie a una velocidad especificada en baudios.
   * **Serial.println(dato):** Imprime datos en el monitor serie seguido de un salto de línea.
   * **delay(ms):** Detiene la ejecución del programa durante un número específico de milisegundos.

**Instrucciones, Componentes y Procedimientos Utilizados:**

**Instrucción 1: Configuración de la Comunicación Serie**

* **Objetivo:** Inicializar la comunicación serie para permitir la transmisión de datos al monitor serie.
* **Sintaxis:**Serial.begin(velocidad);
* **Ejemplo:**Serial.begin(9600);

**Instrucción 2: Obtener el Tiempo Transcurrido**

* **Objetivo:** Utilizar la función **millis()** para obtener el tiempo transcurrido en milisegundos y convertirlo en segundos.
* **Sintaxis:**long tiempo = millis() / 1000;
* **Ejemplo:**long v = millis() / 1000;

**Instrucción 3: Imprimir Datos en el Monitor Serie**

* **Objetivo:** Mostrar el tiempo transcurrido en segundos en el monitor serie.
* **Sintaxis:**Serial.println(dato);
* **Ejemplo:**Serial.println(v);

**Desarrollo:**

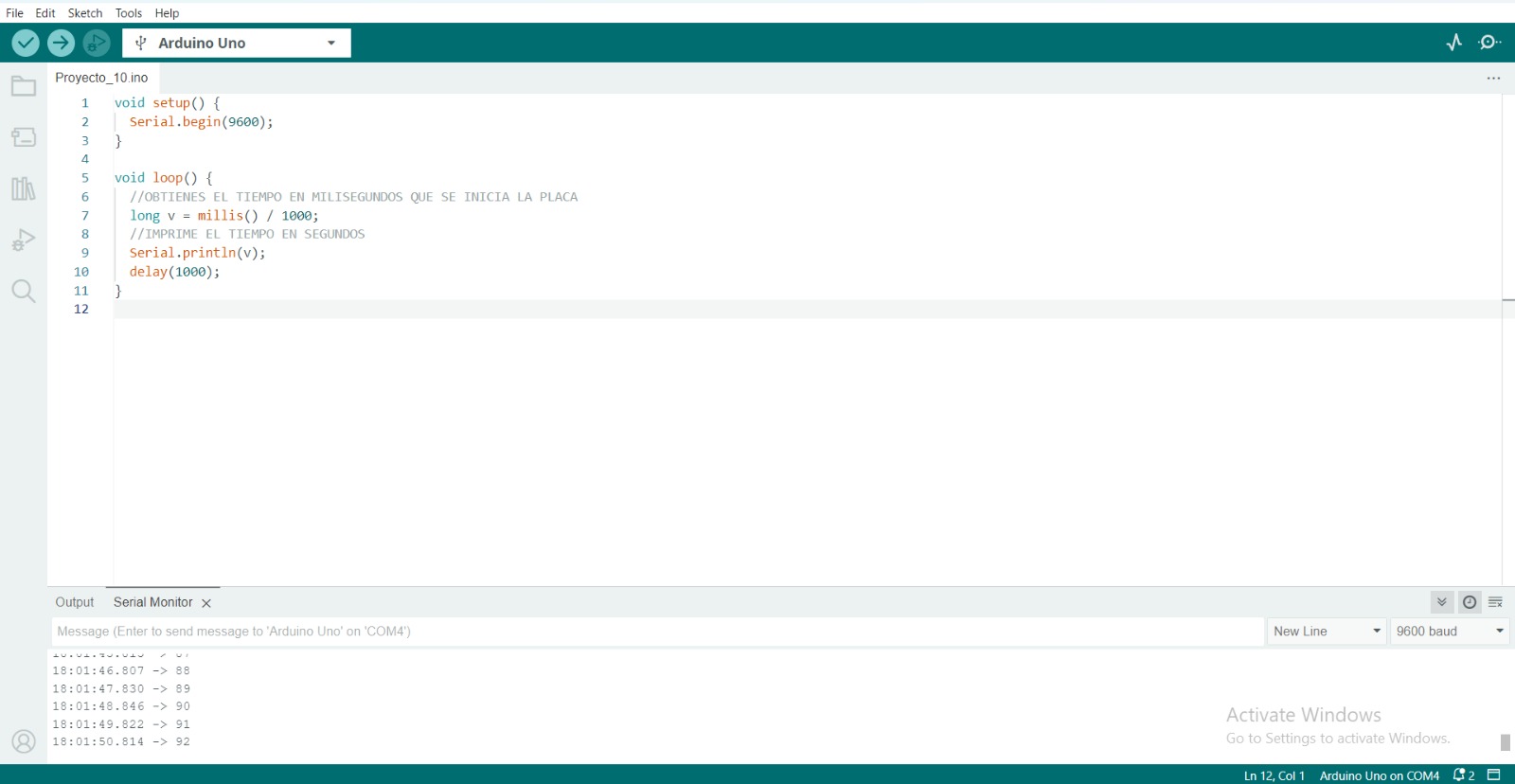
**Paso 1:** Configuración de la Comunicación Serie

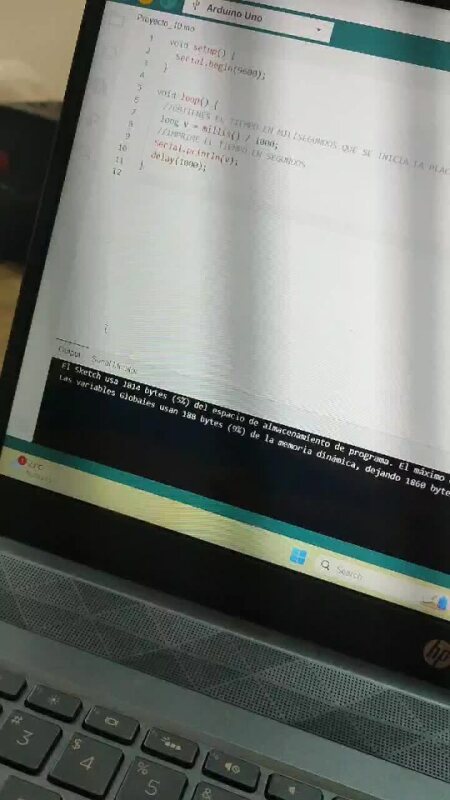
En el setup(), se inicializa la comunicación serie a 9600 baudios utilizando la función Serial.begin.

**Paso 2:** Ciclo Principal (Loop)

En el loop(), se obtiene el tiempo transcurrido en milisegundos desde que se inició la placa Arduino utilizando millis(), se divide por 1000 para convertirlo a segundos y se almacena en la variable v. Luego, se imprime el valor de v en el monitor serie utilizando Serial.println. Finalmente, se utiliza delay(1000) para esperar un segundo antes de repetir el ciclo.

**Demostración:**

****

**[](https://alumnosuatedu-my.sharepoint.com/:v:/g/personal/a2163226011_alumnos_uat_edu_mx/ESJaKcnz6fxEr3uSAGZYEPUBg98lYGxOHVRbQ7aO9O1C1Q?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=NcJCWH)**

**Repositorio(s):**

* **R1.** <https://github.com/Theeduu29/Unidad-1-Embebidos.git>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Nombre del Programa** | **Estado** | **Ubicación** | **Tipo** |
| 1  2  3  4  5  6  7  8  9  10 | Tipos de datos  Encendido y apagado de leds  Print en serial  BlinkLed-seria  Encendido y apagado led buffer  Encendido y apagados leds en ciclo  Encendido y apagado leds iteración  Encendido y apagado por cadena  Encendido y apagado led bucle  Delay de arduino | Completo  Completo  Completo  Completo  Completo  Completo  Completo  Completo  Completo  Completo |  | En Equipo  En Equipo  En Equipo  En equipo  En equipo  En equipo  En equipo  En equipo  En equipo  En equipo |

# Prácticas / Proyectos

**Descripción**

En esta investigación se pretende calcular el presupuesto necesario para la integración de sistemas inteligentes o domóticos en un hogar. La domótica implica el uso de tecnología para automatizar y controlar diversas funciones del hogar, mejorando la eficiencia energética, la seguridad y la comodidad.

**Introducción:**

La integración de sistemas domóticos en el hogar se ha convertido en una tendencia creciente debido a sus múltiples beneficios. Estos sistemas permiten la automatización de tareas rutinarias y el control remoto de dispositivos, lo que resulta en mayor comodidad, seguridad y eficiencia energética.

**Fundamentos Teóricos:**

1. **Domótica:** Conjunto de tecnologías aplicadas al control y la automatización inteligente de la vivienda, que permite una gestión eficiente del uso de la energía y de los recursos.
2. **Componentes Clave:**
   * **Sensores:** Dispositivos que detectan cambios en el entorno (temperatura, movimiento, humedad, etc.).
   * **Actuadores:** Dispositivos que ejecutan acciones (encender luces, abrir puertas, ajustar termostatos).
   * **Controladores:** Unidades centrales que procesan la información de los sensores y envían órdenes a los actuadores.
   * **Interfaces de Usuario:** Aplicaciones o paneles que permiten al usuario interactuar con el sistema.

**Instrucciones, Componentes y Procedimientos Utilizados:**

**Componentes y Costos Estimados:**

**1. Controladores Domóticos:**

* **Objetivo:** Servir como el cerebro del sistema domótico, procesando la información y coordinando los dispositivos.
* **Costo Aproximado:**
  + **Controlador Básico:** $50 - $150
  + **Controlador Avanzado:** $200 - $500

**2. Sensores:**

* **Objetivo:** Detectar condiciones ambientales y cambios en el entorno.
* **Costo Aproximado:**
  + **Sensor de Movimiento:** $15 - $50
  + **Sensor de Temperatura/Humedad:** $10 - $30
  + **Sensor de Puerta/Ventana:** $10 - $20

**3. Actuadores:**

* **Objetivo:** Realizar acciones físicas en respuesta a comandos del controlador.
* **Costo Aproximado:**
  + **Interruptor de Luz Inteligente:** $20 - $50
  + **Enchufe Inteligente:** $15 - $30
  + **Termostato Inteligente:** $100 - $250
  + **Cerradura Inteligente:** $100 - $250

**4. Sistemas de Iluminación Inteligente:**

* **Objetivo:** Controlar y automatizar la iluminación del hogar.
* **Costo Aproximado:**
  + **Bombilla Inteligente:** $10 - $50 por unidad
  + **Kit de Iluminación Inteligente:** $100 - $300

**5. Sistemas de Seguridad Inteligente:**

* **Objetivo:** Proveer seguridad mediante cámaras y sistemas de alarma.
* **Costo Aproximado:**
  + **Cámara de Seguridad Inteligente:** $50 - $200 por unidad
  + **Sistema de Alarma Inteligente:** $100 - $300

**6. Asistentes Virtuales y Hubs:**

* **Objetivo:** Facilitar la interacción con el sistema domótico mediante comandos de voz y aplicaciones.
* **Costo Aproximado:**
  + **Asistente Virtual (Google Home, Amazon Echo):** $50 - $150
  + **Hub Domótico:** $50 - $200

**Desarrollo:**

**Paso 1: Definir Requerimientos y Áreas a Automatizar**

Identificar qué áreas del hogar se quieren automatizar (iluminación, seguridad, climatización, electrodomésticos) y definir los requerimientos específicos para cada una.

**Paso 2: Selección de Componentes**

Elegir los componentes necesarios para cumplir con los requerimientos definidos, considerando la compatibilidad y la integración entre ellos.

**Paso 3: Estimación del Presupuesto**

Realizar una estimación del costo total basado en los componentes seleccionados y su cantidad. A continuación, se presenta un presupuesto estimado para un hogar promedio.

**Presupuesto Estimado:**

**1. Controlador Domótico:**

* Controlador Avanzado: $300

**2. Sensores:**

* 4 Sensores de Movimiento: 4 x $30 = $120
* 4 Sensores de Temperatura/Humedad: 4 x $20 = $80
* 8 Sensores de Puerta/Ventana: 8 x $15 = $120

**3. Actuadores:**

* 10 Interruptores de Luz Inteligente: 10 x $35 = $350
* 5 Enchufes Inteligentes: 5 x $25 = $125
* 2 Termostatos Inteligentes: 2 x $200 = $400
* 2 Cerraduras Inteligentes: 2 x $200 = $400

**4. Sistemas de Iluminación Inteligente:**

* 10 Bombillas Inteligentes: 10 x $25 = $250
* Kit de Iluminación Inteligente: $200

**5. Sistemas de Seguridad Inteligente:**

* 4 Cámaras de Seguridad Inteligente: 4 x $150 = $600
* Sistema de Alarma Inteligente: $200

**6. Asistentes Virtuales y Hubs:**

* 2 Asistentes Virtuales: 2 x $100 = $200
* Hub Domótico: $150

**Total Estimado: $3,495 dls**

**Conclusión:**

La integración de sistemas domóticos en el hogar puede requerir una inversión significativa, pero ofrece numerosos beneficios en términos de comodidad, seguridad y eficiencia energética. El costo total dependerá de la cantidad y calidad de los componentes seleccionados, así como de la complejidad de la instalación.

**Documentación de donde se baso para hacer esta investigación:**

<https://repositorio.upct.es/bitstream/handle/10317/2793/pfc4381.pdf>

**Enlace al Repositorio:**

<https://github.com/Theeduu29/Unidad-1-Embebidos.git>