**Universidad Autónoma de Tamaulipas**

**Facultad de Ingeniería Tampico**

A black background with a black square

Description automatically generated with medium confidenceA red and grey logo

Description automatically generated

**ASIGNATURA**

**Diseño Electrónico Basado en Sistemas Embebidos**

8vo. Semestre – Grupo “G”

2025 -1

**TRABAJO**

**Programas en Clase y Ejercicios**

**UNIDAD**

**1 – Fundamentos Teoricos**

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

|  |  |
| --- | --- |
| **Integrante del Equipo** | **Nivel de Participación** |
| Ortiz Doria Efrain Alejandro | 25% |
| Sandoval Vázquez Christian Michelle | 25% |
| Cruz Bonifacio Luis Fernando | 25% |
| Segura Alonso Adrian | 25% |
| Total: | 100% |

# Índice

[Índice 1](#_Toc188787304)

[Programa: P1 2](#_Toc188787305)

[Programa: P2 3](#_Toc188787306)

[Programa: P3 4](#_Toc188787307)

[Programa: P4 5](#_Toc188787308)

[Programa: P5 6](#_Toc188787309)

[Programa: P6 7](#_Toc188787310)

[Programa: P7 8](#_Toc188787311)

[Programa: P8 9](#_Toc188787312)

[Programa: P9 10](#_Toc188787313)

[Programa: P10 11](#_Toc188787314)

[Programa: P11 12](#_Toc188787315)

[Programa: P12 13](#_Toc188787316)

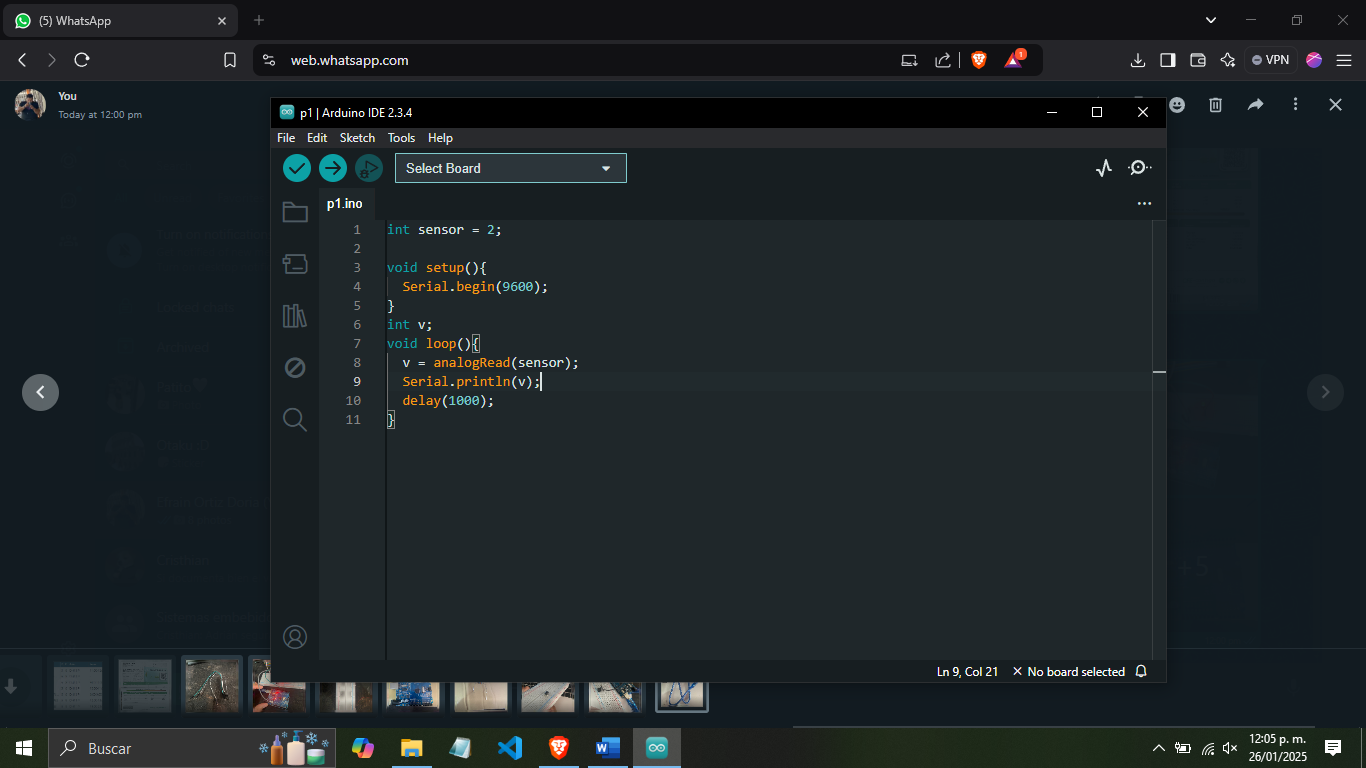
[Fotos 14](#_Toc188787317)

# Programa: P1

1. Funcionalidad  
   Este programa lee un valor analógico desde el pin 2 del Arduino, lo almacena en una variable v y lo imprime en el monitor serial cada segundo.

Imagen sugerida: Diagrama del circuito mostrando un potenciómetro o sensor conectado al pin 2.

1. Instrucciones y estructuras del lenguaje
   * Instrucciones utilizadas:
     + analogRead(sensor): Lee un valor analógico del pin especificado (0 a 1023).
     + Serial.begin(9600): Inicia la comunicación serial a una velocidad de 9600 baudios.
     + Serial.println(v): Envía el valor de v al monitor serial.
     + delay(1000): Introduce un retraso de 1 segundo.
   * Estructura del código:
     + Uso del bucle infinito loop para ejecutar continuamente la lectura e impresión de datos.
2. Variantes para realizar la misma actividad
   * Podría utilizarse analogReadResolution para ajustar la resolución de lectura si la placa lo soporta.
   * Se podrían optimizar los retrasos para usar temporizadores en lugar de delay, lo que permitiría multitarea.
3. Corrección de errores u optimización
   * Errores: El programa funciona correctamente y no presenta errores evidentes.
   * Optimización: Se podría inicializar el pin como entrada explícitamente con pinMode(sensor, INPUT) para mayor claridad.
4. Conclusión y aplicaciones  
   Este programa es ideal para monitorear señales analógicas provenientes de sensores (como un potenciómetro o un sensor de temperatura) y analizar sus valores en tiempo real mediante el monitor serial.
5. Código



# Programa: P2

1. Funcionalidad  
   Este programa lee un valor digital (0 o 1) desde el pin 2 del Arduino y lo imprime en el monitor serial cada segundo.
2. Instrucciones y estructuras del lenguaje
   * Instrucciones utilizadas:
     + digitalRead(sensor): Lee un valor digital (HIGH o LOW) del pin especificado.
     + Serial.begin(9600): Inicia la comunicación serial a 9600 baudios.
     + Serial.println(v): Envía el valor leído al monitor serial.
     + delay(1000): Introduce un retraso de 1 segundo.
   * Estructura del código:
     + Uso del método pinMode(sensor, INPUT) para configurar el pin como entrada.
     + Implementación de un bucle infinito loop para realizar lecturas y mostrar datos continuamente.
3. Variantes para realizar la misma actividad
   * Podría usarse una interrupción en lugar de una lectura en bucle continuo, lo que haría el programa más eficiente al responder únicamente cuando hay un cambio en el estado del sensor.
4. Corrección de errores u optimización
   * Errores: El programa funciona correctamente.
   * Optimización: Se podría incluir una resistencia pull-up o pull-down en el circuito físico si el sensor no la tiene integrada, para evitar lecturas inestables.
5. Conclusión y aplicaciones  
   Este programa es ideal para monitorear el estado de un sensor digital, como un interruptor o un sensor de movimiento, en aplicaciones de domótica o sistemas de control.
6. Codigo

Captura de pantalla de computadora

Descripción generada automáticamente

# Programa: P3

1. Funcionalidad  
   Este programa activa un actuador conectado al pin 10 del Arduino, alternando su estado entre encendido (HIGH) y apagado (LOW) cada segundo.
2. Instrucciones y estructuras del lenguaje
   * Instrucciones utilizadas:
     + digitalWrite(actuador, HIGH): Establece el pin del actuador en estado alto (encendido).
     + digitalWrite(actuador, LOW): Establece el pin del actuador en estado bajo (apagado).
     + delay(1000): Introduce un retraso de 1 segundo entre los cambios de estado.
     + pinMode(actuador, OUTPUT): Configura el pin como salida para controlar el actuador.
   * Estructura del código:
     + El bucle loop alterna continuamente entre encender y apagar el actuador.
3. Variantes para realizar la misma actividad
   * Se podría usar un transistor o un relevador si el actuador requiere más corriente de la que el pin puede proporcionar.
4. Corrección de errores u optimización
   * Errores: El programa es funcional y no presenta errores evidentes.
   * Optimización: Incluir comentarios en el código para describir las acciones realizadas en cada paso.
5. Conclusión y aplicaciones  
   Este programa es ideal para aplicaciones que requieran alternar el estado de un actuador, como encender y apagar un LED o controlar un motor pequeño de manera cíclica.
6. Codigo

Captura de pantalla de computadora

Descripción generada automáticamente

# Programa: P4

1. Funcionalidad  
   Similar al Programa 3, este código activa un actuador conectado al pin 10 del Arduino, alternando entre encendido y apagado cada segundo.

Imagen sugerida: Diagrama de conexión mostrando el actuador controlado por el pin 10.

1. Instrucciones y estructuras del lenguaje
   * Instrucciones utilizadas:
     + Las mismas que en el Programa 3 (digitalWrite, delay, pinMode).
   * Estructura del código:
     + Uso de comentarios para separar las secciones del setup y loop.
2. Variantes para realizar la misma actividad
   * Igual que en el Programa 3, se podría utilizar hardware adicional para actuadores de mayor consumo.
3. Corrección de errores u optimización
   * Errores: El programa es funcional y no presenta errores.
   * Optimización: Se podría incluir una función personalizada para manejar los cambios de estado del actuador.
4. Conclusión y aplicaciones  
   Este programa también es útil para el control básico de actuadores, como en sistemas de iluminación o señalización.
5. Codigo

Captura de pantalla de computadora

Descripción generada automáticamente

# Programa: P5

1. Funcionalidad  
   Este programa genera una señal PWM en el pin 10, aumentando y disminuyendo gradualmente el ciclo de trabajo, lo que podría utilizarse para controlar la intensidad de un LED o la velocidad de un motor.
2. Instrucciones y estructuras del lenguaje
   * Instrucciones utilizadas:
     + analogWrite(pinpwm, i): Genera una señal PWM con un ciclo de trabajo ajustable entre 0 y 255.
     + delayMicroseconds(10): Introduce un retraso muy corto entre iteraciones para suavizar la transición.
   * Estructura del código:
     + Uso de bucles for para incrementar y luego decrementar el valor del ciclo de trabajo.
3. Variantes para realizar la misma actividad
   * Podría usarse una librería como SoftPWM para generar señales PWM en pines que no son PWM por defecto.
   * La velocidad de cambio del ciclo de trabajo podría ajustarse dinámicamente según entradas externas, como un potenciómetro.
4. Corrección de errores u optimización
   * Errores: No se inicializa el pin como salida usando pinMode(pinpwm, OUTPUT), lo que podría causar problemas en algunas placas.
   * Optimización: Incrementar el valor del retraso en delayMicroseconds para hacer el efecto más visible en hardware real.
5. Conclusión y aplicaciones  
   Este programa es ideal para aplicaciones donde se requiera variar la intensidad de un LED o controlar la velocidad de motores DC mediante modulación por ancho de pulso (PWM).
6. Codigo

Captura de pantalla de computadora

Descripción generada automáticamente

# Programa: P6

1. Funcionalidad  
   Este programa mide el tiempo transcurrido desde que se inició el Arduino utilizando la función millis() y lo imprime en el monitor serial cada segundo.
2. Instrucciones y estructuras del lenguaje
   * Instrucciones utilizadas:
     + millis(): Devuelve el tiempo en milisegundos desde que comenzó a ejecutarse el programa.
     + Serial.println(v): Imprime el valor de la variable v en el monitor serial.
     + delay(1000): Introduce un retraso de 1 segundo.
   * Estructura del código:
     + Uso de la función millis() dentro del bucle loop para obtener el tiempo transcurrido.
3. Variantes para realizar la misma actividad
   * Podría usarse micros() para medir el tiempo en microsegundos, si se requiere mayor precisión.
4. Corrección de errores u optimización
   * Errores: El programa no presenta errores funcionales.
   * Optimización: Agregar comentarios para describir el propósito de cada sección del código.
5. Conclusión y aplicaciones  
   Este programa es útil para medir tiempos en proyectos que requieran monitoreo o sincronización precisa, como cronómetros o temporizadores.
6. Codigo

Captura de pantalla de computadora

Descripción generada automáticamente

# Programa: P7

1. Funcionalidad  
   Este programa lee un valor analógico desde el pin 2, lo escala utilizando la función map a un rango de 0 a 255 y genera una señal PWM proporcional en el pin 10.
2. Instrucciones y estructuras del lenguaje
   * Instrucciones utilizadas:
     + analogRead(sensor): Lee un valor analógico entre 0 y 1023.
     + map(valor, 0, 1023, 0, 255): Escala el valor leído al rango de 0 a 255.
     + analogWrite(actuador, v): Genera una señal PWM proporcional al valor escalado.
   * Estructura del código:
     + Lectura, escalado y escritura del valor dentro del bucle loop.
3. Variantes para realizar la misma actividad
   * Usar una ecuación matemática en lugar de map para realizar el escalado.
4. Corrección de errores u optimización
   * Errores: El programa es funcional y no presenta errores.
   * Optimización: Configurar explícitamente los pines como entrada y salida con pinMode.
5. Conclusión y aplicaciones  
   Este programa es útil en aplicaciones que requieran control proporcional, como la regulación de intensidad de luz o velocidad de motores en función de una entrada analógica.
6. Codigo

Captura de pantalla de computadora

Descripción generada automáticamente

# Programa: P8

1. Funcionalidad  
   Este programa lee un valor digital desde el pin 2, lo divide entre 4 para obtener un rango reducido y genera una señal PWM en el pin 10 proporcional al valor obtenido.
2. Instrucciones y estructuras del lenguaje
   * Instrucciones utilizadas:
     + digitalRead(sensor): Lee un valor digital (0 o 1).
     + analogWrite(actuador, v): Genera una señal PWM proporcional al valor calculado.
     + map(valor, 0, 1, 0, 255): Ajusta el rango del valor digital a un rango proporcional de 0 a 255.
   * Estructura del código:
     + Lectura del valor digital, escalado mediante la función map y escritura del valor ajustado al actuador mediante PWM.
3. Variantes para realizar la misma actividad
   * Podría usarse un sensor digital diferente que entregue valores altos o bajos según las condiciones del entorno, como un sensor de proximidad o un botón.
4. Corrección de errores u optimización
   * Errores: El programa es funcional y no presenta errores evidentes.
   * Optimización: Se podría incluir un tiempo de espera (delay) para evitar lecturas muy rápidas que podrían generar inconsistencias.
5. Conclusión y aplicaciones  
   Este programa es ideal para aplicaciones que requieran una respuesta proporcional a una entrada binaria, como la intensidad de un LED según un interruptor.
6. Codigo

Captura de pantalla de computadora

Descripción generada automáticamente

# Programa: P9

1. Funcionalidad  
   Este programa utiliza un botón conectado al pin 2 como entrada digital para controlar un LED conectado al pin 10, encendiéndolo y apagándolo cada vez que se presiona el botón.
2. Instrucciones y estructuras del lenguaje
   * Instrucciones utilizadas:
     + digitalRead(pinboton): Lee el estado del botón (HIGH o LOW).
     + digitalWrite(pinled, HIGH/LOW): Establece el estado del LED según la lógica del programa.
     + Uso de una variable de estado para alternar el encendido y apagado del LED.
   * Estructura del código:
     + Configuración inicial de los pines como entrada (botón) y salida (LED).
     + Implementación de lógica de alternancia utilizando el estado anterior del botón.
3. Variantes para realizar la misma actividad
   * Usar una interrupción en lugar de un bucle para detectar el cambio de estado del botón.
   * Implementar un botón adicional para apagar el LED directamente.
4. Corrección de errores u optimización
   * Errores: Podría presentarse rebote en el botón, lo que generaría lecturas incorrectas. Esto se puede solucionar con un retardo de debounce.
   * Optimización: Añadir una función de debounce para eliminar lecturas erróneas del botón.
5. Conclusión y aplicaciones  
   Este programa es útil para aplicaciones que requieran control manual de dispositivos, como interruptores de luz o controladores básicos de hardware.
6. Codigo

Captura de pantalla de computadora

Descripción generada automáticamente

# Programa: P10

1. Funcionalidad  
   Este programa genera una secuencia de encendido y apagado en cuatro LEDs conectados a los pines 8, 9, 10 y 11, simulando el efecto de una "carrera" de luces.
2. Instrucciones y estructuras del lenguaje
   * Instrucciones utilizadas:
     + digitalWrite(pinled, HIGH/LOW): Enciende o apaga cada LED según la secuencia definida.
     + delay(tiempo): Introduce un retardo para visualizar el efecto de carrera.
   * Estructura del código:
     + Configuración de los pines como salidas.
     + Uso de un bucle para alternar el encendido de cada LED en orden.
3. Variantes para realizar la misma actividad
   * Cambiar la secuencia de encendido para simular otros efectos, como encendido simultáneo o aleatorio.
   * Usar LEDs RGB para variar colores en la secuencia.
4. Corrección de errores u optimización
   * Errores: No se detectan errores evidentes.
   * Optimización: Implementar un retardo más corto para hacer la animación más rápida, o controlar el tiempo mediante un potenciómetro.
5. Conclusión y aplicaciones  
   Este programa es ideal para aplicaciones decorativas o de señalización, como efectos visuales en sistemas de luces LED.
6. Codigo.

Captura de pantalla de computadora

Descripción generada automáticamente

# Programa: P11

1. Funcionalidad  
   Este programa utiliza un potenciómetro conectado al pin A0 para controlar la velocidad de encendido y apagado de un LED conectado al pin 10.
2. Instrucciones y estructuras del lenguaje
   * Instrucciones utilizadas:
     + analogRead(potenciometro): Lee el valor del potenciómetro (0 a 1023).
     + map(valor, 0, 1023, minimo, maximo): Ajusta el rango de valores para controlar el tiempo de retardo.
     + digitalWrite(pinled, HIGH/LOW): Enciende y apaga el LED.
   * Estructura del código:
     + Lectura continua del potenciómetro.
     + Uso del valor leído para ajustar dinámicamente el retardo entre los cambios de estado del LED.
3. Variantes para realizar la misma actividad
   * Controlar varios LEDs simultáneamente con diferentes velocidades según entradas adicionales.
   * Usar un sensor de luz en lugar de un potenciómetro para variar la velocidad según la iluminación ambiental.
4. Corrección de errores u optimización
   * Errores: No se inicializa explícitamente el pin del LED como salida.
   * Optimización: Implementar suavizado del valor leído del potenciómetro para evitar fluctuaciones en la velocidad.
5. Conclusión y aplicaciones  
   Este programa es útil para aplicaciones interactivas que requieran un control sencillo de velocidad, como en sistemas de iluminación dinámica.
6. Codigo

Captura de pantalla de computadora

Descripción generada automáticamente

# Programa: P12

1. Funcionalidad  
   Este programa utiliza dos botones conectados a los pines 2 y 3 para aumentar y disminuir la intensidad de un LED conectado al pin 10, utilizando modulación por ancho de pulso (PWM).
2. Instrucciones y estructuras del lenguaje
   * Instrucciones utilizadas:
     + digitalRead(pinboton): Lee el estado de los botones.
     + analogWrite(pinled, intensidad): Ajusta la intensidad del LED según el valor de la variable intensidad.
     + Uso de condiciones if para incrementar o disminuir el valor de la intensidad.
   * Estructura del código:
     + Configuración inicial de los pines como entradas (botones) y salida (LED).
     + Control de límites para la variable de intensidad (0 a 255).
3. Variantes para realizar la misma actividad
   * Usar un encoder rotativo en lugar de botones para un control más fluido de la intensidad.
   * Implementar un botón adicional para restablecer la intensidad al valor inicial.
4. Corrección de errores u optimización
   * Errores: Podría presentarse rebote en los botones, lo que generaría ajustes erróneos en la intensidad.
   * Optimización: Implementar debounce por hardware (capacitor) o software (retardo).
5. Conclusión y aplicaciones  
   Este programa es ideal para aplicaciones donde se requiera un control preciso de intensidad, como en sistemas de iluminación o interfaces de usuario.
6. Codigo

Captura de pantalla de computadora

Descripción generada automáticamente

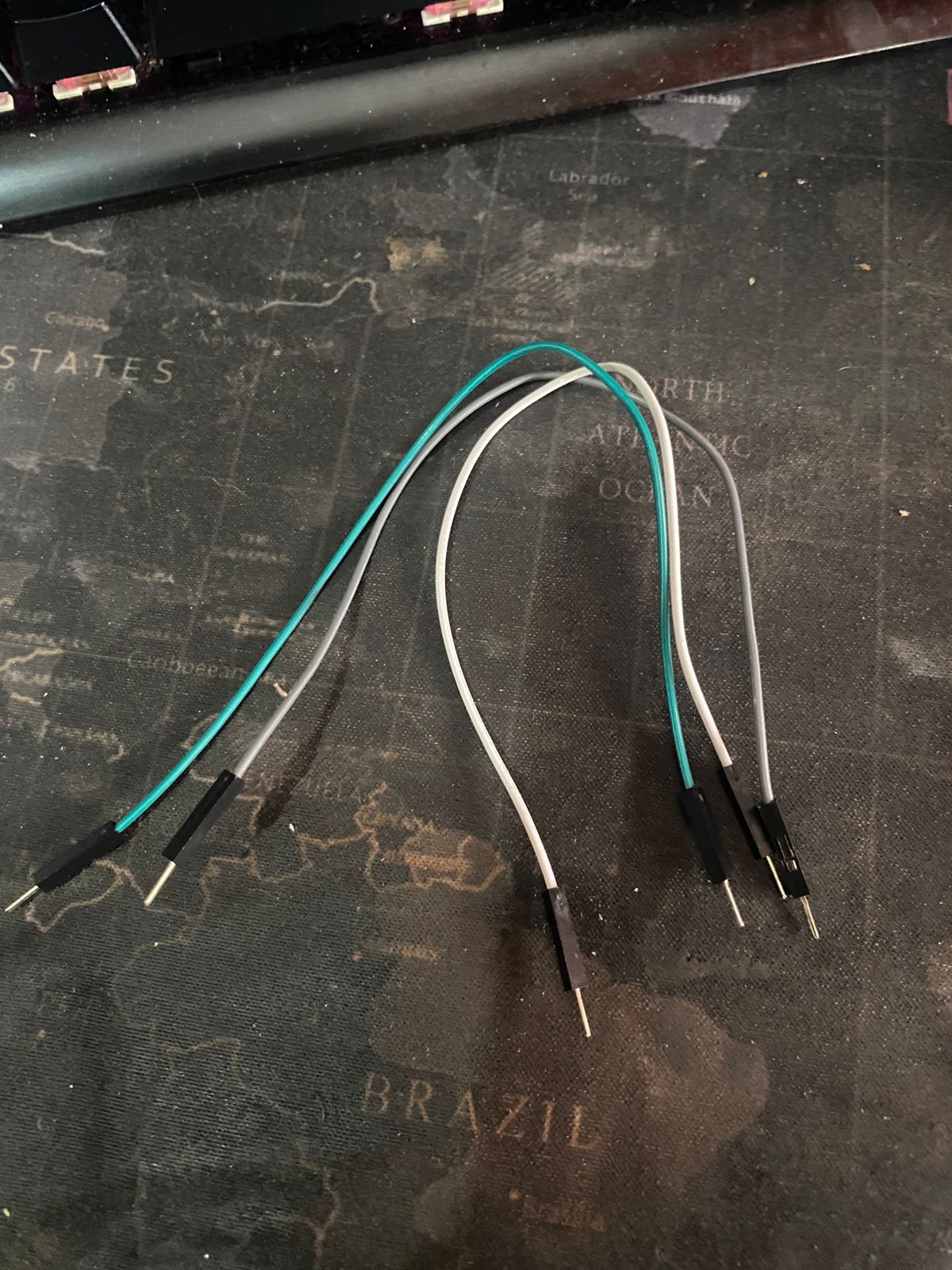
# Fotos

Un circuito electrónico

Descripción generada automáticamente con confianza baja

Imagen que contiene reloj, parado, cuarto

Descripción generada automáticamenteUn circuito electrónico

Descripción generada automáticamente con confianza mediaUn teclado de computadora

Descripción generada automáticamente con confianza mediaImagen que contiene interior, computadora, escritorio, tabla

Descripción generada automáticamenteImagen que contiene bicicleta, computadora, tabla, teclado

Descripción generada automáticamenteImagen que contiene pequeño, tabla, puesto, bolsa

Descripción generada automáticamente