Módulo 4: Programación funcional.

módulos y librerías.

- cómo usar los módulos
- cómo se instalar librerias

Estructura de programación.

- funciones.
- máquina de estados.

ejercicios:

Uso de ultrasonido
Sensor de humedad y temperatura
Imprimir los datos en el LCD

Implementar una mef que para controlar el robot:

-Tome los datos de temperatura y humedad.
-Tome datos de un sensor de ultrasonido.
-Responder a las señales que lleguen por serial.
‘w’ avanzar.
‘s’ retroceder.
‘a’ rotar a la izquierda.
‘d’ rotar a la izquierda.
‘p’ cerrar pinza.
‘P’ abrir pinza.
‘c’ subir cámara.
‘C’ bajar cámara.

Módulo 4: Programación funcional.

módulos

Existen una amplia variedad de módulos disponibles en el mercado, cada uno diseñado para cumplir una función específica.

Por ejemplo, puede ser un sensor de temperatura, movimiento, sensor de luz, un módulo Bluetooth, GPS, un motor, una pantalla LCD, relés, etc.

Los módulos y sensores son dispositivos que se utilizan junto con microcontroladores o sistemas embebidos para expandir sus capacidades y permitir la interacción con el entorno. Estos módulos suelen estar diseñados como placas compactas que contienen un componente principal, como un **sensor** o un **actuador**, junto con el circuito necesario para su funcionamiento adecuado. Regularmente este circuito se encarga de alimentar correctamente los componentes, acondicionar las señales y hasta podría tener un indicador lumínico por ejemplo.

Estos módulos y sensores ofrecen una forma conveniente de agregar funcionalidades específicas a un proyecto, evitando la necesidad de diseñar y construir circuitos desde cero. Facilitan la integración y aceleran el desarrollo de prototipos y proyectos, ya que proporcionan una interfaz estandarizada y listos para usar con los microcontroladores.

Los sensores capturan datos del entorno.
Los actuadores realizan acciones físicas.

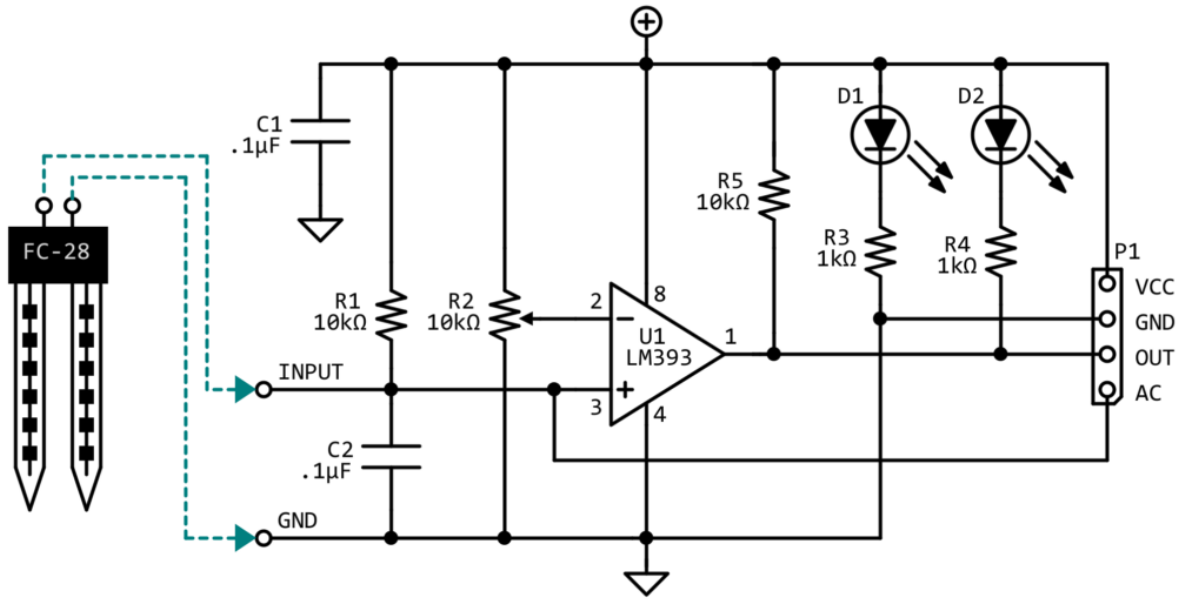
El uso de sensores en un desarrollo no solo simplifica los circuitos también, prescindiendo de algunas prestaciones pueden simplificar la programación.

La imagen anterior es de un módulo sensor de humedad de suelo.
Este módulo tiene dos pines que se conectan al electrodo y por el otro lado tiene 4, VCC (5v), GND, D0 y A0.
Por lo que podemos tomar un valor analogico con el pin A0 hacer calculos y tomar decisiones, o definir un valor umbral con el preset (cuadrado celeste con una circulo blanco y una x) y usar la salida digital como un una entrada digital (HIGH o LOW).

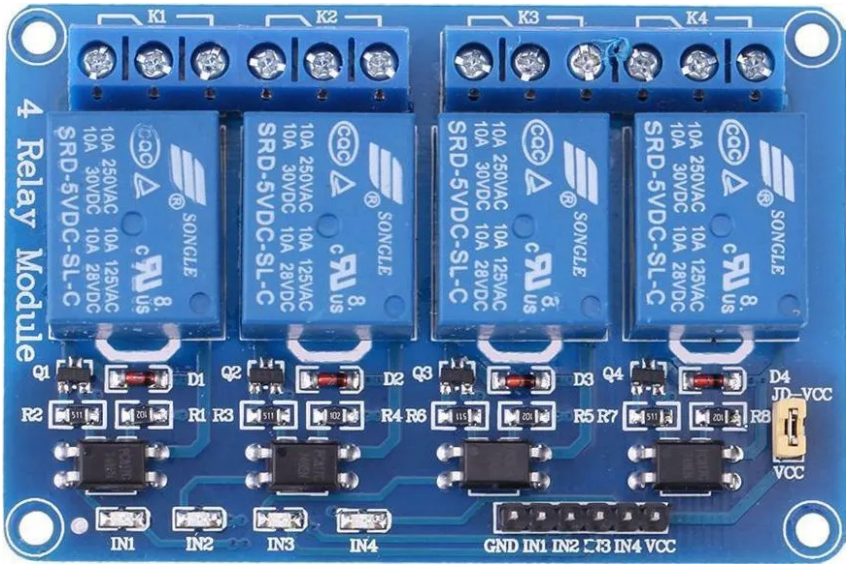
A modo informativo analizamos el circuito del módulo.

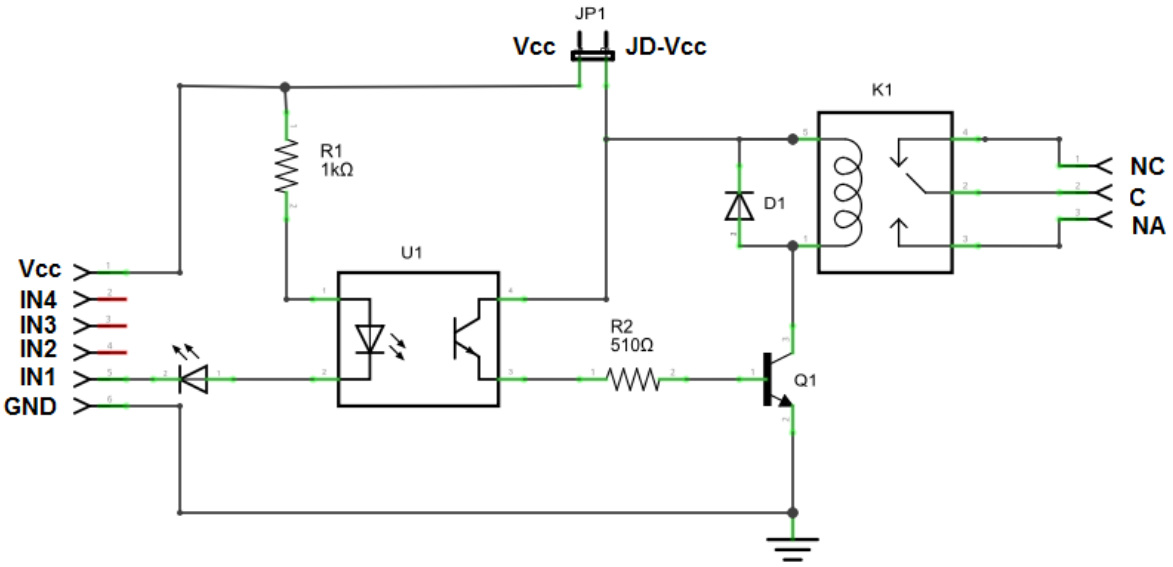
Si optamos por usar la salida analógica, AC vemos que está conectada a R1 y C2 estando C2 en paralelo con el sensor, si ignoramos el capacitor, es el mismo arreglo que usamos para conectar el LDR. Este capacitor está para estabilizar la medición, si recordamos, no se puede cargar ni descargar instantáneamente. D1 es un led que solo indica que el módulo está alimentado. C1 es un capacitor de desacoplamiento.

Si por el contrario usamos la salida digital. OUT tenemos que ajustar el preset (R2) al nivel de conmutación que nos interese. Si se sumerge el electrodo en agua, donde la humedad sería máxima, la resistencia de este baja, por lo que a la salida del comparador conectada a OUT será LOW encendiendo el led D2.



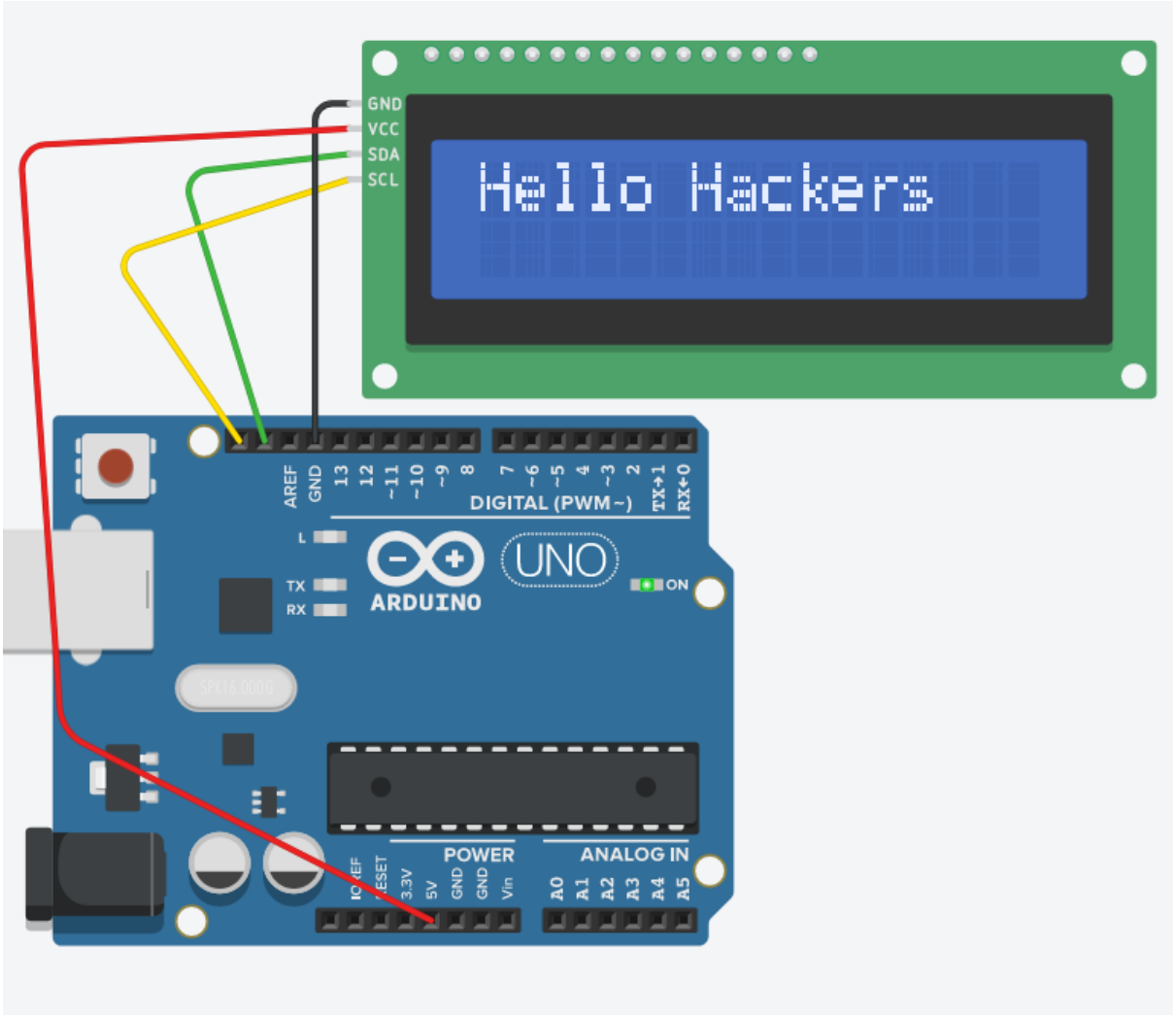
Otro módulo que nos resuelve la etapa circuital, por ejemplo a la hora de encender una bomba de agua para regar el césped es uno conocido como módulo relé.
Este módulo ofrece la solución de etapa de potencia y a su vez un circuito de aislación.





Existen variedad de Módulos con diferente complejidad y para distintos usos, por ahora solo los usamos con señales analógicas (conectando a un pin A0, A1, A…) o digitales en cualquier pin, preguntando si el estado es una señal alta o baja. En sensores que brindan otras mediciones como temperatura, presión, aceleración o intensidad de campo magnético la variable en sí es más compleja por que no alcanza con conocer el estado de un pin o la lectura de un analógico.
Estos tipos de sensores tendrán que conectarse a un bus de comunicación y puede involucrar varios pines.

Librerías.



Las librerías en arduino son conjuntos de código predefinido que contienen funciones y rutinas listas para usar, diseñadas para facilitar la programación y la interacción con componentes y periféricos específicos. Estas librerías proporcionan un conjunto de funciones y métodos que simplifican tareas comunes, como la comunicación con sensores, el control de servos, la manipulación de pantallas LCD, el manejo de comunicación inalámbrica y muchas otras funcionalidades.

En Arduino, las librerías son archivos de código que se pueden importar en un proyecto para aprovechar las funciones y características que ofrecen. Estas librerías ahorran tiempo y esfuerzo, ya que proporcionan una abstracción de alto nivel de la complejidad subyacente y permiten a los programadores utilizar funcionalidades avanzadas sin tener que entender todos los detalles internos.
¿Esto es bueno, malo?

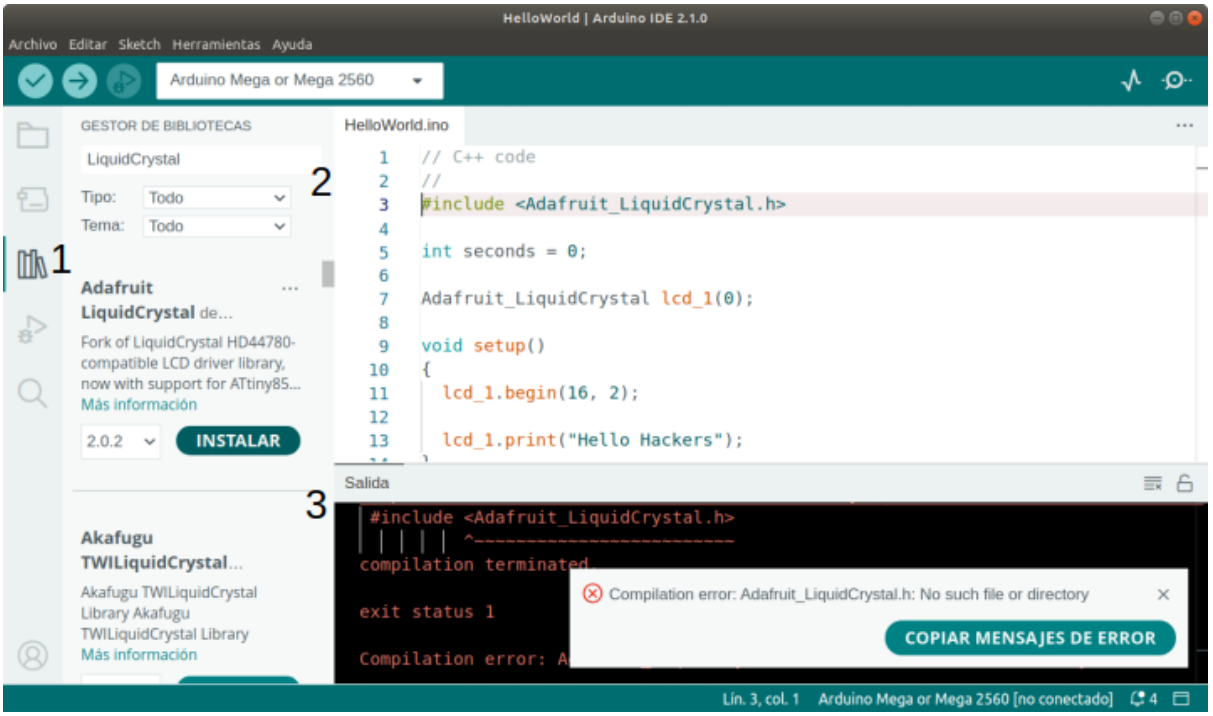
Arduino IDE incluye una amplia biblioteca estándar de librerías que cubren una variedad de funciones básicas y periféricos comunes. Además, hay una gran cantidad de librerías desarrolladas por la comunidad Arduino que abarcan una amplia gama de aplicaciones y componentes específicos.

Para utilizar una librería en un proyecto de Arduino, generalmente se deben seguir los siguientes pasos:

1. Descargar la librería: Si la librería que se necesita no está incluida en la biblioteca estándar de Arduino IDE, se debe descargar e instalar la librería específica desde fuentes confiables. Esto se puede hacer desde el administrador de librerías incorporado en Arduino IDE o mediante la descarga manual desde el repositorio correspondiente.
2. Importar la librería: Una vez descargada, la librería se debe importar en el proyecto. Esto se hace mediante la opción "Incluir librería" en el menú "Sketch" de Arduino IDE.
3. Utilizar las funciones de la librería: Una vez importada la librería, se pueden utilizar las funciones y métodos proporcionados por la misma en el código del proyecto. Esto se logra llamando a las funciones relevantes dentro del programa.

Las librerías en Arduino simplifican la programación al proporcionar bloques de código reutilizables y probados. Ayudan a los programadores a evitar tener que escribir todo el código desde cero y permiten un desarrollo más rápido y eficiente de proyectos. Además, la comunidad Arduino es muy activa, lo que significa que hay una gran cantidad de librerías disponibles y probadas por muchas personas, para casi cualquier funcionalidad que se necesite.

Al instalar la librería también encontramos ejemplos nuevos de los que podemos ir aprendiendo a usarla.



Para descargar la librería
Vamos al gestor de librerías **(1)**
Escribimos el nombre **(2)**
Instalar **(3)**

Estructura de programación.

Funciones

Para definir una función en Arduino, se siguen los siguientes pasos:

Declaración de la función: La declaración de la función se coloca antes de la función `void setup()` y `void loop()` en el código de Arduino. La declaración especifica el tipo de retorno de la función, el nombre de la función y los parámetros que recibe (si los hay). Por ejemplo:

```
tipo_retorno nombre_funcion(parametros);

bool saludo(void);
```

El tipo_retorno es el tipo de dato que devuelve la función, como void si no devuelve ningún valor o puede ser int, float, etc. El nombre_funcion es el nombre que le das a la función como la vamos a utilizar después y los parametros son las variables que recibe la función (opcional).

Implementación de la función: Después de la declaración, se debe escribir el cuerpo de la función, que contiene las instrucciones que se ejecutarán cuando la función sea llamada. El cuerpo de la función se coloca entre llaves {}. Por ejemplo:

```
bool saludo(void){
      lcd_1.setBacklight(1);
      lcd_1.setCursor(0, 1);
      lcd_1.print("Chau Hackers");
      return true; // no tiene sentido devolver un valo
}
```

la usamos como cualquier función solo con su nombre, en este ejemplo está declarada como void por lo que no requiere parámetros. Cada vez que se ejecute la línea `saludo();` en el programa. se imprimirá, si todo es correcto, “Chau Hackers” en el lcd.
si seguimos estos pasos la función es global. por lo que la podemos llamar en cualquier lado, incluso hacer funciones que usen funciones.

Máquina de estados.

Las máquinas de estado son modelos conceptuales utilizados en programación y electrónica para representar el comportamiento de un sistema que puede estar en diferentes estados. Las transiciones entre ellos en respuesta a eventos o condiciones específicas.
Una máquina de estado está compuesta por un conjunto de estados, eventos y acciones.

- Estados: Representan las condiciones o situaciones en las que se puede encontrar el sistema en un momento dado. Cada estado tiene un nombre descriptivo y puede tener asociadas acciones específicas que se realizan cuando se ingresa o sale de ese estado.
- Eventos: Son sucesos o condiciones que desencadenan una transición de un estado a otro. Pueden ser eventos físicos, como la presión de un botón, o eventos internos, como la finalización de una tarea, fin de un cronómetro, etc. Los eventos determinan qué transiciones se activan en la máquina de estado.
- Transiciones: Son los cambios de estado que ocurren en respuesta a un evento. Cada transición está definida por el estado de origen, el evento que la desencadena y el estado de destino. Además, puede haber condiciones o acciones asociadas a una transición, que se evalúan o realizan al realizar la transición.
- Acciones: Son las tareas o actividades que se ejecutan cuando se ingresa o sale de un estado o cuando se realiza una transición. Las acciones pueden ser simples, como encender un LED, o más complejas, como iniciar una secuencia de operaciones.

Las máquinas de estado son útiles para modelar sistemas que tienen comportamientos distintos en diferentes situaciones y necesitan responder a eventos de manera específica. Permiten organizar el código de manera modular y estructurada, facilitando el mantenimiento y comprensión del sistema. Además, brindan un enfoque claro para la lógica de control y permiten gestionar de manera efectiva las interacciones entre estados y eventos.

En el contexto de Arduino, las máquinas de estado son especialmente útiles para el desarrollo de proyectos en los que se requiere una gestión precisa de los estados y eventos, como controladores de sistemas, interfaces de usuario o aplicaciones interactivas.

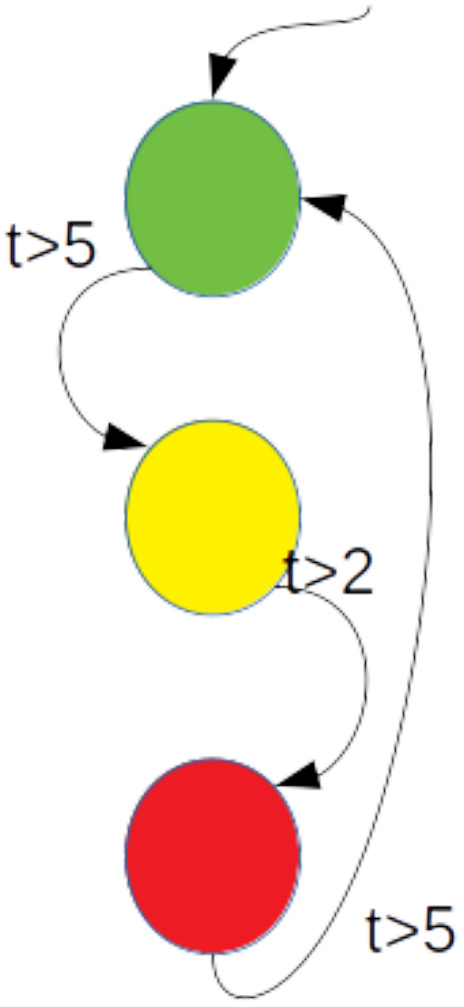
Ejemplo:

La máquina de estados de un semáforo es un ejemplo clásico de aplicación de este concepto. Un semáforo consta de diferentes estados que representan las diferentes señales de tráfico, como luz verde, luz amarilla y luz roja, y transiciones que ocurren en respuesta a eventos, como cambios en el tiempo.

- Estado Verde: En este estado, la luz verde está encendida, lo que indica que los vehículos pueden avanzar. La duración de este estado puede ser definida por un temporizador.
- Transición 1: Cuando se cumple el tiempo asignado para el estado verde, se produce una transición al siguiente estado.

- Estado Amarillo: En este estado, la luz amarilla está encendida, lo que indica una advertencia de que el semáforo está a punto de cambiar a rojo. La duración de este estado puede ser breve.
- Transición 2: Después de que se cumpla el tiempo asignado para el estado amarillo, se produce una transición al siguiente estado.
- Estado Rojo: En este estado, la luz roja está encendida, lo que indica que los vehículos deben detenerse. La duración de este estado puede ser más larga.
- Transición 3: Si se detecta un evento, como presionar un botón de cruce de peatones, se produce una transición que puede alterar el comportamiento normal del semáforo, permitiendo el paso seguro de peatones.
- Transición 4: Después de que se cumpla el tiempo asignado para el estado rojo, se produce una transición que vuelve al estado verde, reiniciando el ciclo del semáforo.

La máquina de estados del semáforo se puede implementar utilizando estructuras de control condicionales, como `if-else` o `switch-case`, y temporizadores para controlar la duración de los estados. Además, se pueden utilizar sensores o pulsadores para activar las transiciones según sea necesario.



```
const int pinLedVerde = 2;
const int pinLedAmarillo = 3;
const int pinLedRojo = 4;

enum Estado { VERDE, AMARILLO,ROJO};

Estado estadoActual = VERDE;

void encenderLedVerde(void);
void encenderLedAmarillo(void);
void encenderLedRojo(void);

void setup() {
 pinMode(pinLedVerde, OUTPUT);
 pinMode(pinLedAmarillo, OUTPUT);
 pinMode(pinLedRojo, OUTPUT);
}



void loop() {
 switch (estadoActual) {
   case VERDE:
     encenderLedVerde();
     delay(5000);
     estadoActual = AMARILLO;
     break;


   case AMARILLO:
     encenderLedAmarillo();
     delay(2000);
     estadoActual = ROJO;
     break;

   case ROJO:
     encenderLedRojo();
     delay(5000);
     estadoActual = VERDE;
     break;
 }
}
```

```
void encenderLedVerde() {
 digitalWrite(pinLedVerde, HIGH);
 digitalWrite(pinLedAmarillo, LOW);
 digitalWrite(pinLedRojo, LOW);
}

void encenderLedAmarillo() {
 digitalWrite(pinLedVerde, LOW);
 digitalWrite(pinLedAmarillo, HIGH);
 digitalWrite(pinLedRojo, LOW);
}

void encenderLedRojo() {
 digitalWrite(pinLedVerde, LOW);
 digitalWrite(pinLedAmarillo, LOW);
 digitalWrite(pinLedRojo, HIGH);
}
```

ejercicios:
Uso de ultrasonido

Sensor de humedad y temperatura

Recuperar los códigos anteriores y Imprimir los datos en el LCD

Implementar una mef que para controlar el robot:

-Tome los datos de temperatura y humedad.
-Tome datos de un sensor de ultrasonido.
-Responder a las señales que lleguen por serial.
‘w’ avanzar.
‘s’ retroceder.
‘a’ rotar a la izquierda.
‘d’ rotar a la izquierda.
‘p’ cerrar pinza.
‘P’ abrir pinza.
‘c’ subir cámara.
‘C’ bajar cámara.