METODOS MAS UTILIZADOS EN ARDUINO:

Lista de 30 funciones o métodos importantes y ampliamente utilizados en Arduino, junto con una breve explicación de cada uno:

pinMode(**pin**, **mode**): Establece el modo de un pin en específico, ya sea como entrada (INPUT) o salida (OUTPUT).

digitalRead(pin): Lee el estado lógico (alto o bajo) de un pin digital.

digitalWrite(**pin**, **value**): Escribe un estado lógico (alto o bajo) en un pin digital.

analogRead(**pin**): Lee el valor analógico (0-1023) de un pin analógico.

analogWrite(**pin**, **value**): Escribe un valor de señal analógica (PWM) en un pin digital.

delay(ms): Detiene la ejecución del programa durante un período de tiempo especificado en milisegundos.

millis(): Devuelve el número de milisegundos transcurridos desde que Arduino se inició.

Serial.**begin**(**baudRate**): Inicia la comunicación serial con una velocidad de baudios específica.

Serial.**print**(**data**): Envía datos a través del puerto serial como texto legible.

Serial.**read**(): Lee un byte de datos recibidos a través del puerto serial.

Serial.**available**(): Devuelve el número de bytes disponibles para la lectura en el búfer del puerto serial.

random(**min**, **max**): Genera un número pseudoaleatorio dentro de un rango especificado.

map(**value**, **fromLow**, **fromHigh**, **toLow**, **toHigh**): Mapea un valor desde un rango original a un nuevo rango.

attachInterrupt(**digitalPin**, **ISR**, **mode**): Asocia una función de interrupción (ISR) a un pin digital específico.

detachInterrupt(**digitalPin**): Desactiva la función de interrupción asociada a un pin digital.

EEPROM.**write**(**address**, **value**): Escribe un byte de datos en la memoria EEPROM.

EEPROM.**read**(**address**): Lee un byte de datos de la memoria EEPROM.

Wire.**begin**(): Inicia la comunicación I2C (bus de dos cables).

Wire.**beginTransmission**(**address**): Inicia una transmisión de datos hacia un dispositivo I2C específico.

Wire.**write**(**data**): Escribe datos en el búfer de transmisión para enviar a través de I2C.

Wire.**endTransmission**(): Finaliza la transmisión I2C y envía los datos almacenados en el búfer.

Wire.**requestFrom**(**address**, **quantity**): Solicita una cantidad específica de bytes desde un dispositivo I2C.

SPI.**begin**(): Inicia la comunicación SPI (Interfaz de Periféricos en Serie).

SPI.**transfer**(**data**): Envía un byte de datos a través de SPI y devuelve el byte recibido.

SPI.**beginTransaction**(**settings**): Inicia una transacción SPI con configuraciones específicas.

SPI.**endTransaction**(): Finaliza la transacción SPI actual.

Servo.attach(**pin**): Asocia un pin a un objeto de servo para controlar un motor servo.

Servo.**write**(**angle**): Establece el ángulo objetivo para un motor servo.

Ethernet.**begin**(**mac**): Inicia la conexión Ethernet utilizando una dirección MAC específica.

EthernetClient.**connect**(**server**, **port**): Establece una conexión TCP con un servidor remoto en una dirección IP y puerto determinados.

Estas son solo algunas de las muchas funciones y métodos disponibles en Arduino. Su utilidad varía según el proyecto específico que estés desarrollando.

BIBLIOTECAS PARA ARDUINO:

Aquí tienes una lista de algunas de las bibliotecas más utilizadas en Arduino, junto con una breve explicación de cada una:

Wire: Esta biblioteca proporciona funciones para la comunicación en bus I2C (Inter-Integrated Circuit). Permite a Arduino actuar como maestro o esclavo en una red I2C, lo que facilita la conexión de múltiples dispositivos.

SPI: La biblioteca SPI permite la comunicación utilizando el protocolo SPI (Serial Peripheral Interface). Se utiliza principalmente para la comunicación con periféricos y sensores que admiten este protocolo, como pantallas TFT, acelerómetros, tarjetas SD, etc.

Servo: Esta biblioteca simplifica el control de los motores servo. Permite controlar la posición y el movimiento de los servos conectados a los pines PWM del Arduino.

Ethernet: La biblioteca Ethernet facilita la conexión y comunicación con redes Ethernet. Permite a Arduino actuar como cliente o servidor en aplicaciones de red, como el control de dispositivos a través de Internet.

WiFi: Esta biblioteca permite la conexión a redes Wi-Fi utilizando los módulos Wi-Fi compatibles con Arduino. Proporciona funciones para configurar y administrar conexiones Wi-Fi, enviar/recibir datos y controlar servicios en la nube.

LiquidCrystal: Esta biblioteca es especialmente útil para controlar pantallas LCD de caracteres alfanuméricos. Permite mostrar texto y controlar el cursor en pantallas LCD basadas en el controlador Hitachi HD44780.

Adafruit NeoPixel: Esta biblioteca es utilizada para controlar tiras de LED programables NeoPixel. Permite controlar cada LED individualmente para crear efectos de iluminación personalizados.

Adafruit Sensor: Esta biblioteca proporciona una interfaz unificada para acceder a los datos de los sensores. Es comúnmente utilizada junto con las bibliotecas específicas de los sensores de Adafruit para facilitar la lectura de datos de una amplia variedad de sensores.

DHT: Esta biblioteca permite la lectura de datos de los sensores de temperatura y humedad DHT11, DHT21 y DHT22. Proporciona funciones para leer los valores de temperatura y humedad ambiental.

EEPROM: La biblioteca EEPROM simplifica la lectura y escritura de datos en la memoria EEPROM interna del Arduino. Permite almacenar datos de forma persistente incluso después de que se apague el Arduino.

Estas son solo algunas de las muchas bibliotecas disponibles para Arduino. La elección de la biblioteca adecuada depende del proyecto y los componentes específicos que estés utilizando.