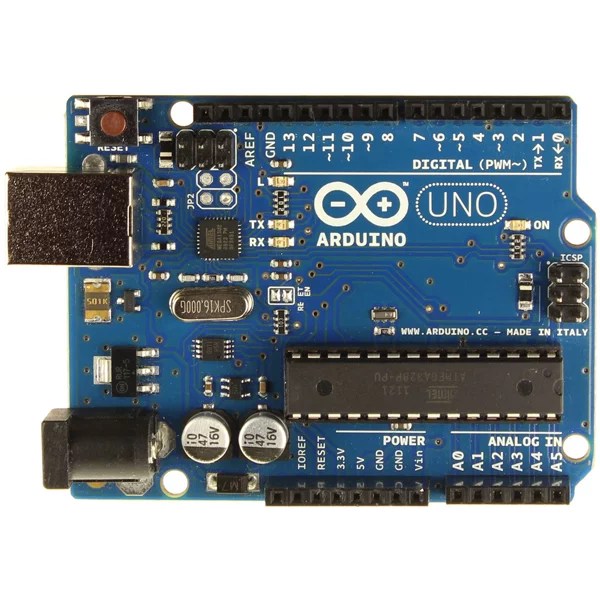
**Funcionamiento del Código Iot: Arduino Uno**

1. **Importación de Librerías:** Utilizamos librerías como Wire para la comunicación I2C, LiquidCrystal\_I2C para controlar una pantalla LCD, y SPI para la comunicación SPI. Estas librerías permiten la interacción con diversos componentes de hardware.
2. **Declaración de Variables y Configuración Inicial:** Se declaran variables para almacenar datos y se configuran los pines de los motores y sensores. Por ejemplo, se definen los pines para los motores y el sensor LDR, así como las variables para almacenar los valores leídos de estos componentes.
3. **Inicialización en setup():** En la función setup(), se configuran los pines como entradas o salidas, se inicializan las comunicaciones SPI y la pantalla LCD, y se muestra un mensaje de inicio en la pantalla LCD. También se inicia la comunicación serial para la depuración.
4. **Control de Motores:** Se crean funciones como setOutputMotor1 y setOutputMotor2 para controlar los motores paso a paso. Estas funciones configuran los pines de los motores según los valores de las tablas de búsqueda lookupDerecha y lookupIzquierda para mover los motores en la dirección deseada.
5. **Movimiento de Motores:** Se implementan funciones como MoverMotorDerecha, MoverMotorIzquierda, MoverMotor2Derecha y MoverMotor2Izquierda para mover los motores un número específico de pasos en la dirección indicada. Estas funciones utilizan las tablas de búsqueda y las funciones de control de motores para realizar el movimiento.
6. **Funciones de Reconocimiento:** Se desarrollan varias funciones de reconocimiento (ReconocimientoV, ReconocimientoEsp, ReconocimientoH) que leen valores del sensor LDR y mueven los motores para tomar lecturas en diferentes posiciones. Estas funciones permiten analizar el entorno y detectar patrones específicos.
7. **Cálculo de Temperatura:** La función DatoTemperatura utiliza la comunicación SPI para leer datos de un sensor de temperatura y calcular la temperatura en grados Celsius. Esta función convierte los datos leídos en un valor de temperatura utilizable.
8. **Lectura de Valores del LDR:** La función printLDRValue lee el valor del sensor LDR y lo devuelve. Esta función se utiliza en varias partes del código para obtener lecturas del sensor de luz.
9. **Búsqueda de Mínimos:** La función findMinIndex encuentra el índice del valor mínimo en un arreglo. Esta función se utiliza para determinar la posición con la menor lectura del sensor, lo que puede indicar la presencia de un objeto o una condición específica.
10. **Funciones de Ejecución (E1, E2, E3, E4):** Estas funciones mueven los motores a posiciones específicas y leen valores del sensor LDR y del sensor de temperatura. Se utilizan para realizar acciones específicas basadas en las lecturas de los sensores.
11. **Funciones de Manejo de Patrones (H1 a H8):** Estas funciones combinan las funciones de reconocimiento y ejecución para analizar patrones complejos. Mueven los motores a diferentes posiciones, realizan lecturas y ejecutan acciones basadas en los resultados.
12. **Bucle Principal (loop()):** En la función loop(), se verifica el estado de un interruptor y se ejecutan las funciones de reconocimiento y manejo de patrones según el estado del interruptor. Si el interruptor está activado, se realiza el reconocimiento y se ejecutan las acciones correspondientes. Si no, se mueve el motor a una posición de mantenimiento y se muestra un mensaje en la pantalla LCD.



**Funcionamiento del Código Iot: ESP8266**

1. **Configuración de la Conexión Serial:**

Se utiliza %serialconnect to --port=COM3 --baud=115200 para establecer la conexión serial con el puerto COM3 a una velocidad de 115200 baudios.

1. **Importación de Módulos:**

Se importan los módulos UART y time de la biblioteca machine para manejar la comunicación serial y las pausas temporales.

1. **Inicialización del Puerto Serial:**

Se configura el puerto serial UART con una velocidad de 115200 baudios mediante uart = UART(0, baudrate=115200).

1. **Bucle Principal:**

Se entra en un bucle infinito while True: para mantener la ejecución continua del programa.

1. **Verificación de Datos Disponibles:**

Se verifica si hay datos disponibles en el puerto serial con if uart.any():.

1. **Lectura de Datos:**

Si hay datos disponibles, se leen con data = uart.readline(). Esta función lee una línea completa de datos del puerto serial.

1. **Decodificación de Datos:**

Se intenta decodificar los datos recibidos de bytes a texto utilizando data\_str = data.decode('utf-8').strip(). Esto convierte los datos a una cadena de texto y elimina los espacios en blanco al inicio y al final.

1. **Procesamiento de Datos:**

Los datos decodificados se dividen en una lista utilizando datos = data\_str.split(','), separando los elementos por comas.

1. **Validación de Datos:**

Se asegura que la lista de datos tenga exactamente tres elementos antes de proceder a imprimirlos. Esto se hace asignando los valores a d1, d2, y d3 y verificando if len(datos) == 3:.

1. **Importación de Módulos:**  
   Se importan los módulos network, time, ustruct, urandom y MQTTClient de umqtt.simple para manejar la conexión Wi-Fi, la generación de números aleatorios y la comunicación MQTT.
2. **Configuración de la Red Wi-Fi:**

Se definen las credenciales de la red Wi-Fi con SSID y PASSWORD.

1. **Configuración del Servidor MQTT:**

Se configuran los parámetros del servidor MQTT, incluyendo la dirección IP (mqtt\_server), el puerto (mqtt\_port), el ID del cliente (mqtt\_id) y el tema (mqtt\_topic).

1. **Configuración del Motor de Coordenadas:**

Se define una lista de posiciones del motor (motor\_positions) que incluye ángulos desde 0 hasta 360 grados en incrementos de 45 grados.

1. **Función conectar\_wifi():**

Esta función activa la interfaz Wi-Fi, intenta conectarse a la red especificada y espera hasta que la conexión se establezca. Imprime la configuración de la red una vez conectada.

1. **Función main():**

Llama a conectar\_wifi() para establecer la conexión Wi-Fi.

Configura y conecta al servidor MQTT utilizando MQTTClient.

En un bucle continuo, simula datos de sensores con simular\_datos(), publica estos datos en el tema MQTT configurado y espera 3 segundos antes de enviar el siguiente conjunto de datos.

Maneja excepciones durante la conexión y el envío de datos, imprimiendo cualquier error que ocurra.

1. Finalmente, desconecta del servidor MQTT.

Un circuito electrónico

Descripción generada automáticamente con confianza media

**Funcionamiento del Código en Python:**

1. **Importación de Módulos:**

Se importan los módulos paho.mqtt.client para la comunicación MQTT, datetime para manejar fechas y horas, Workbook de openpyxl para trabajar con archivos Excel, y os para operaciones del sistema.

1. **Variables Globales:**

Se declaran variables globales dato1, dato2, y dato3 para almacenar los datos recibidos.

1. **Ruta al Archivo Excel:**

Se define la ruta del archivo Excel con file\_path = "datos\_estacion.xlsx".

1. **Función crear\_excel():**

Esta función verifica si el archivo Excel existe. Si no existe, crea un nuevo libro de trabajo y una hoja de trabajo, agrega encabezados en la primera fila y guarda el archivo. Si el archivo ya existe, imprime un mensaje indicando esto.

1. **Función guardar\_datos\_en\_excel():**

Esta función abre el archivo Excel existente, agrega una nueva fila con los datos recibidos (fecha, hora, LDR, temperatura, coordenada del motor) y guarda los cambios.

1. **Callback on\_connect():**

Esta función se ejecuta cuando el cliente se conecta al broker MQTT. Imprime un mensaje de conexión y se suscribe al tema “helion”.

1. **Callback on\_message():**

Esta función se ejecuta cuando se recibe un mensaje del broker MQTT. Decodifica el mensaje recibido, lo divide en tres datos (LDR, temperatura, coordenada del motor) y los asigna a las variables globales. Luego, obtiene la fecha y hora actuales, imprime los datos recibidos y los guarda en el archivo Excel.

1. **Creación del Archivo Excel:**

Se llama a crear\_excel() para asegurarse de que el archivo Excel exista antes de comenzar a recibir datos.

1. **Configuración del Cliente MQTT:**

Se configura el cliente MQTT, asignando las funciones on\_connect y on\_message como callbacks.

1. **Conexión al Broker MQTT:**

Se conecta al broker MQTT en la dirección IP 192.168.0.5 y puerto 1883.

1. **Ejecución del Bucle Principal de MQTT:**

Se ejecuta client.loop\_forever() para mantener el cliente en funcionamiento y procesar los mensajes entrantes de manera continua.

**Funcionamiento de PowerBi:**

PowerBi se conecta a la base de datos local creada por el programa en Python para poder hacer una interfaz gráfica y un análisis de los datos recolectados por el dispositivo Iot.

Interfaz de usuario gráfica

Descripción generada automáticamente

**Cotización apróximada:**

* Jumper 10 cm MxM, 10 Piezas (2 unidades): Q7.00
* Jumper 20 cm MxH, 10 Piezas (1 unidad): Q8.00
* Jumper 10 cm MxH, 10 Piezas (3 unidades): Q7.00
* Pantalla LCD 1602, Azul con I2C (1 unidad): Q50.00
* Protoboard de 1/2 galleta (1 unidad): Q29.00
* Motor de pasos 28BYJ-48-5V (2 unidades): Q29.00
* Módulo controlador de stepper ULN2003 (2 unidades): Q18.00
* Módulo regulador de voltaje LM2596 de 5V 3A con dos puertos USB (1 unidad): Q29.00
* Protector de acrílico para Uno R3, negro (1 unidad): Q19.00
* Panel solar de 12V a 1.5W (1 unidad): Q79.00
* Switch ON OFF SPDT, negro rectangular 6A (1 unidad): Q9.00
* Módulo de foto resistencia analógico y digital (1 unidad): Q19.00
* Arduino UNO + Wifi ESP8266: Q155.00
* Arduino UNO: Q138.00
* Sensor de temperatura MAX6675: Q130.00

**Total: Q756.00**