¡Claro! Vamos a desglosar el código paso a paso para que puedas entenderlo en detalle:

---

1. Clase Base: `Sensor`

Objetivo:

La clase base `Sensor` define las características comunes a todos los sensores, como los atributos y métodos esenciales.

Atributos:

```
protected:
    float valor;       // Almacena el valor medido por el sensor.
    String magnitud;   // Describe qué mide el sensor (e.g., "Temperatura", "Luz").
```

- **`protected`:** Permite que las clases derivadas accedan a estos atributos.

Constructor:

cpp
Copiar código

```
Sensor(String magnitudInicial) : valor(0.0), magnitud(magnitudInicial) {}
```

- Inicializa `valor` en 0.0 y asigna una descripción al sensor mediante `magnitudInicial`.

Métodos:

cpp
Copiar código

```
virtual float leerMedicion(int pin) = 0;
```

- **`virtual` y `= 0`:** Declara un método **virtual puro**, lo que significa que debe ser implementado por las clases derivadas. Esto hace que `Sensor` sea una **clase abstracta**.

```
virtual void mostrarMedicion();
```

- Muestra el valor y la magnitud del sensor en el monitor serial.

---

2. Clase Derivada: `SensorTemperatura`

Objetivo:

Simula un sensor de temperatura, como el **TMP36**.

Constructor:

```
SensorTemperatura() : Sensor("Temperatura") {}
```

- Llama al constructor de la clase base y establece la magnitud como "Temperatura".

Método `leerMedicion`:

```
float SensorTemperatura::leerMedicion(int pin) {
    int lectura = analogRead(pin);
    valor = (lectura * 5.0 / 1023.0 - 0.5) * 100;
    return valor;
}
```

- Lee un valor analógico del pin `A0` (0-1023).
- Convierte ese valor a grados Celsius usando la fórmula del TMP36.

---

3. Clase Derivada: `SensorLuz`

Objetivo:

Simula un sensor de luz con un fotoresistor.

Constructor:

```
SensorLuz() : Sensor("Luminosidad") {}
```

- Llama al constructor base y establece la magnitud como "Luminosidad".

Método `leerMedicion`:

```
float SensorLuz::leerMedicion(int pin) {
    int lectura = analogRead(pin);
    valor = (lectura * 100.0 / 1023.0);
    return valor;
}
```

- Lee un valor analógico del pin `A1`.
- Convierte la lectura a un porcentaje de luminosidad (0% a 100%).

---

4. Clase Derivada: `SensorUltrasonido`

Objetivo:

Simula un sensor de distancia basado en ultrasonido, como el **HC-SR04**.

Constructor:

```
```

```
SensorUltrasonido(int tPin, int ePin) : Sensor("Distancia"), trigPin(tPin), echoPin(ePin)
{}
```

- Llama al constructor base con la magnitud "Distancia".
- Inicializa los pines **Trig** y **Echo**.

Método `leerMedicion`:

```
float SensorUltrasonido::leerMedicion(int pin) {
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    long duration = pulseIn(echoPin, HIGH);
    valor = duration * 0.034 / 2;
    return valor;
}
```

1. Envía un pulso al pin **Trig**.
2. Mide el tiempo que tarda el eco en regresar al pin **Echo**.
3. Calcula la distancia en centímetros usando la fórmula `distancia = tiempo * velocidad_sonido / 2`.

---

5. Programa Principal

Definición de Sensores:

```
SensorTemperatura sensorTemp;
SensorLuz sensorLuz;
SensorUltrasonido sensorUltra(8, 9);
```

- Crea un sensor de temperatura, uno de luz y uno de ultrasonido.
- Para el ultrasonido, se especifican los pines **Trig** (8) y **Echo** (9).

`setup`: Inicialización

```
void setup() {
    Serial.begin(9600);

    pinMode(8, OUTPUT); // Configuración del ultrasonido
    pinMode(9, INPUT);
}
```

- Configura el monitor serial y los pines necesarios para el sensor de ultrasonido.

`loop`: Lectura y Mostrar Mediciones

```
void loop() {
    // Sensor de temperatura
    float temp = sensorTemp.leerMedicion(A0);
    sensorTemp.mostrarMedicion();

    // Sensor de luz
    float luz = sensorLuz.leerMedicion(A1);
    sensorLuz.mostrarMedicion();

    // Sensor de ultrasonido
    float distancia = sensorUltra.leerMedicion(0); // pin no usado
    sensorUltra.mostrarMedicion();

    delay(2000);
}
```

1. Sensor de Temperatura:

```
float temp = sensorTemp.leerMedicion(A0);
sensorTemp.mostrarMedicion();
```

- Lee la temperatura del **TMP36** conectado al pin **A0**.
- Muestra el valor y la magnitud.

2. Sensor de Luz:

```
float luz = sensorLuz.leerMedicion(A1);
sensorLuz.mostrarMedicion();
```

- Lee la luz desde el fotoresistor conectado al pin **A1**.
- Muestra el valor y la magnitud.

3. Sensor de Ultrasonido:

```
float distancia = sensorUltra.leerMedicion(0);
sensorUltra.mostrarMedicion();
```

- Mide la distancia usando los pines configurados.
- Muestra el valor y la magnitud.

Resumen

1. **Diseño modular:** Las clases permiten manejar diferentes tipos de sensores con facilidad.
2. **Reutilización:** La clase base define las propiedades comunes, mientras que las clases derivadas implementan las diferencias específicas.
3. **Adaptabilidad:** Es fácil agregar más sensores creando nuevas clases que hereden de `Sensor`.