# UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

*Universidad Del Perú, Decana de América*

**Facultad de Ingeniería de Sistemas e Informática**

Escuela Académica Profesional de Ingeniería de Software


## Curso: Internet de las cosas

## Docente: Yessica Rosas

## Estudiante:

Caballero Leon, Fredi Alexander
Reyna Ulloa, Eduardo Juan
Suarez Palomino, Jean Paul

# 2023

### DESCRIPCIÓN DEL PROYECTO

Un sistema automatizado para abrir y cerrar cortinas basado en la luz solar y controlado remotamente mediante una plataforma IoT, utilizando un microcontrolador ESP32/ESP8266. Incorpora sensores de luz (LDR), temperatura, y proximidad para detectar condiciones ambientales, y actuadores como un motor de corriente continua (DC) con engranajes, un driver de motor (L298N) y un LCD para mostrar la temperatura.

La plataforma Blynk permite el control remoto, mientras que el sistema se alimenta con un adaptador de corriente USB y una fuente específica para el motor. Adicionalmente, se incluyen cables, conectores, poleas y cuerda para el movimiento de las cortinas.

**Monitoreo de temperatura**

Además de la necesidad de luz, los seres humanos también requieren un ambiente térmico confortable para sus actividades diarias. Por eso, el control de la temperatura en un hogar u oficina es fundamental para asegurar el bienestar y la eficiencia energética.

- Importancia de la Temperatura en el Sistema Automatizado de Cortinas

El sistema automatizado de cortinas no solo se basa en la intensidad de la luz solar para operar, sino que también tiene en cuenta la temperatura ambiente. Esto permite optimizar la entrada de luz solar y el aislamiento térmico, asegurando un entorno cómodo y eficiente en términos de energía.

- Parámetros de Temperatura

Temperatura base:

Es la temperatura promedio diaria mínima que se considera confortable para los ocupantes del espacio. Si la temperatura cae por debajo de este valor, el sistema puede cerrar las cortinas para retener el calor.

Temperatura óptima:

Es la mejor temperatura promedio diaria para mantener un ambiente agradable y confortable. El sistema puede ajustar las cortinas para maximizar la entrada de luz solar cuando la temperatura se encuentra dentro de este rango, aprovechando el calor natural.

Temperatura máxima:

Es la temperatura máxima que se considera confortable. Por encima de este valor, el sistema cierra las cortinas para bloquear la entrada de calor y mantener una temperatura interior agradable.

**Iluminación Ambiental**

Además de la temperatura, la luz natural es un factor crucial para el confort y la eficiencia energética en los espacios interiores. Un sistema automatizado de cortinas debe tener en cuenta la intensidad de la luz solar para optimizar la iluminación natural, reduciendo la dependencia de la luz artificial y mejorando el bienestar de los ocupantes.

- Importancia de la Iluminación Ambiental en el Sistema Automatizado de Cortinas

El control de la iluminación ambiental permite ajustar automáticamente las cortinas para aprovechar al máximo la luz natural disponible, mejorando la eficiencia energética y creando un ambiente interior agradable. Esto es especialmente importante en oficinas y hogares, donde la calidad de la luz puede afectar la productividad y el estado de ánimo de las personas.

- Parámetros de Iluminación

Nivel de luz mínima:

Es el nivel de iluminación mínimo necesario para que el espacio sea funcional sin la necesidad de luz artificial adicional. Si la intensidad de la luz solar cae por debajo de este nivel, el sistema puede abrir las cortinas para dejar entrar más luz natural.

Nivel de luz óptima:

Es el nivel de iluminación que proporciona el mejor equilibrio entre luz natural y confort visual. En este rango, las cortinas se ajustan para maximizar el uso de la luz solar sin causar deslumbramiento o incomodidad.

Nivel de luz máxima:

Es el nivel de iluminación por encima del cual la luz solar puede causar deslumbramiento o sobre iluminación. El sistema cierra las cortinas parcialmente para reducir el exceso de luz y evitar la incomodidad visual.

**SOLUCIÓN (avance)**

**Componentes**

- ESP32
- Sensor ultrasónico (HC-SR04)
- Sensor de luz (LDR)
- Sensor de temperatura y humedad (DHT22)
- Driver de motor (L298N)
- Motor de corriente continua (DC)
- Pantalla LCD (20x4)

**Monitoreo de iluminación**

- **Descripción del Sensor y su Configuración:**

El sistema utiliza un sensor LDR (Light Dependent Resistor) digital para monitorear la intensidad de la luz ambiental. El LDR está conectado al pin digital 34 del microcontrolador ESP32.

El LDR detecta si la luz ambiental está por encima o por debajo de un umbral predefinido (LIGHT_THRESHOLD), lo que se indica con un valor digital de ALTO (HIGH) o BAJO (LOW).

En el bucle principal (loop()), se lee el valor digital del LDR usando digitalRead(LDR_DO_PIN).

- **Lectura y Procesamiento de Datos:**

Este valor se utiliza para determinar si las cortinas deben abrirse o cerrarse, en combinación con las lecturas de temperatura y humedad.

- **Control de Cortinas:**

Si el valor digital del LDR es ALTO (indicando alta intensidad de luz) o si las condiciones de temperatura o humedad superan ciertos umbrales, se llama a la función openCurtain() para abrir las cortinas.

Si el valor es BAJO y las condiciones de temperatura y humedad están dentro de los límites, se llama a la función closeCurtain() para cerrar las cortinas.

**Monitoreo de temperatura**

- **Descripción del Sensor y su Configuración:**

El sistema utiliza un sensor DHT22 para medir la temperatura y la humedad ambiental. El sensor está conectado al pin 19 del ESP32 y se inicializa con la biblioteca DHT.

El DHT22 proporciona lecturas precisas de temperatura y humedad que se actualizan en cada iteración del bucle principal.

- **Lectura y Procesamiento de Datos:**

En el bucle principal (loop()), se leen los valores de temperatura y humedad utilizando dht.readTemperature() y dht.readHumidity(), respectivamente.

Estos valores se muestran en una pantalla LCD I2C (LiquidCrystal_I2C) para proporcionar información en tiempo real al usuario.

- **Control de Cortinas:**

Si la temperatura leída supera el umbral de temperatura (TEMP_THRESHOLD) o la humedad supera el umbral de humedad (HUMIDITY_THRESHOLD), se llama a la función openCurtain() para abrir las cortinas.

Si las lecturas de temperatura y humedad están por debajo de los umbrales definidos, se llama a la función closeCurtain() para cerrar las cortinas.

**Accionador**

- **Descripción del Motor y su Configuración:**

El sistema utiliza un motor paso a paso controlado por un driver A4988. Los pines de paso (STEP_PIN) y dirección (DIR_PIN) del motor están conectados a los pines 16 y 17 del ESP32.

La biblioteca AccelStepper se utiliza para controlar el motor paso a paso, permitiendo ajustes de velocidad y aceleración.

- **Control del Movimiento del Motor:**

La función openCurtain() establece el pin de dirección (DIR_PIN) en ALTO y realiza 200 pasos para abrir las cortinas, utilizando digitalWrite(STEP_PIN, HIGH) y digitalWrite(STEP_PIN, LOW) con retardos entre cada paso.

La función closeCurtain() establece el pin de dirección en BAJO y realiza 200 pasos para cerrar las cortinas de manera similar.

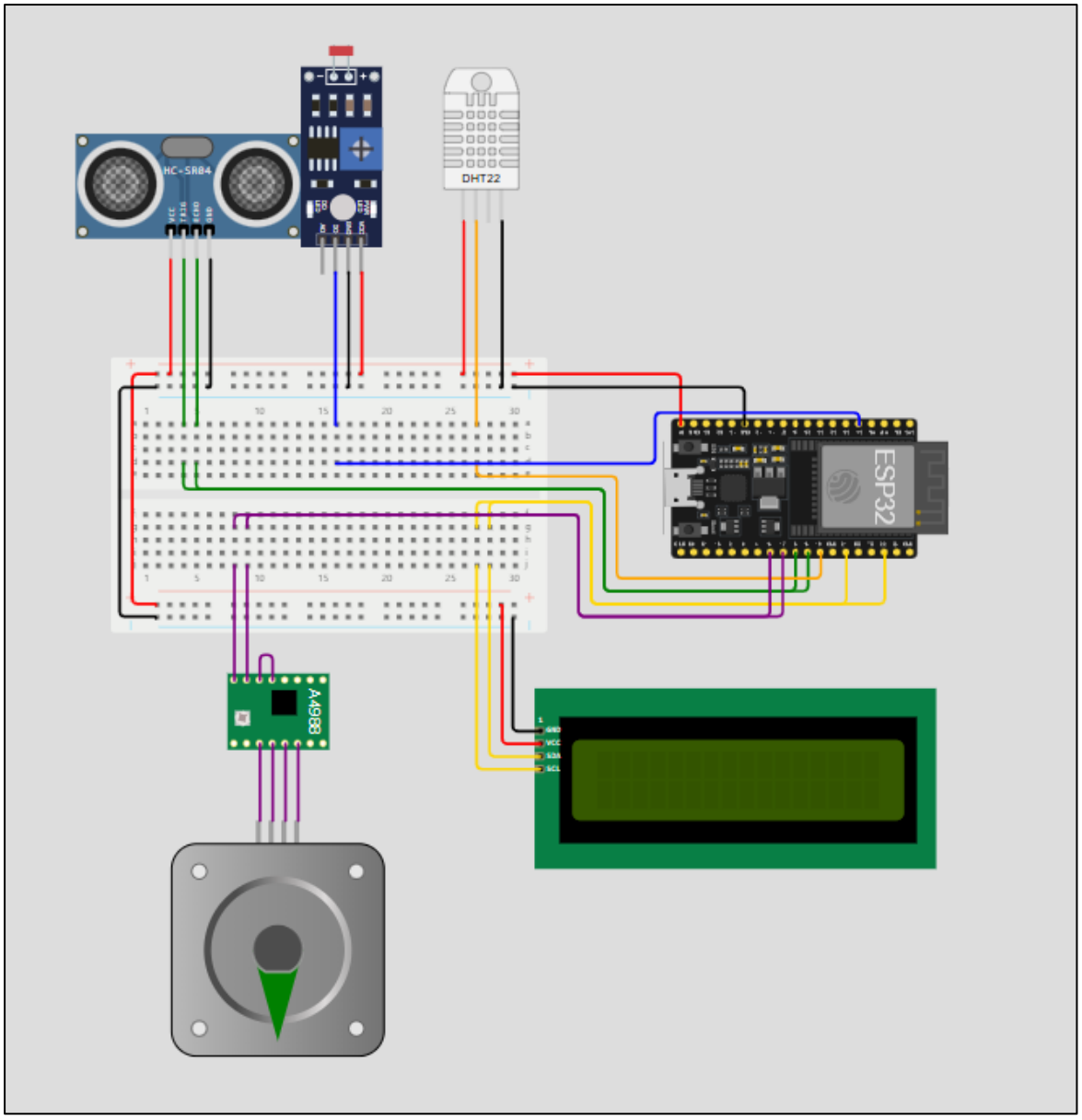
- **Detección de Estado de las Cortinas:**

El sistema utiliza un sensor ultrasónico HC-SR04 para medir la distancia y determinar si las cortinas están completamente abiertas o cerradas.

Se envía un pulso de disparo (TRIG_PIN) y se mide el tiempo de retorno del eco (ECHO_PIN) para calcular la distancia.

Si la distancia medida indica que las cortinas están completamente abiertas (distancia mayor a 10 cm) o completamente cerradas (distancia menor a 5 cm), se actualiza el estado de la bandera isCurtainOpen.

## VISUALIZACIÓN DE LA SIMULACIÓN




**CÓDIGO:**

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
```

```
#include <AccelStepper.h>

#define DHTPIN 19
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 20, 4); // Dirección I2C del LCD y tamaño (20x4)

// Driver de motor paso a paso A4988
#define STEP_PIN 16
#define DIR_PIN 17

AccelStepper stepper(AccelStepper::DRIVER, STEP_PIN, DIR_PIN);

// Pin analógico del LDR
#define LDR_AO_PIN 34

// Pines del sensor ultrasónico HC-SR04
#define TRIG_PIN 5
#define ECHO_PIN 18

// Umbrales para temperatura, humedad y luz
#define TEMP_THRESHOLD 25.0 // Ajustar según necesidad
#define HUMIDITY_THRESHOLD 70.0 // Ajustar según necesidad
#define LIGHT_THRESHOLD 2000 // Ajustar según necesidad (1024 es el máximo
valor en una lectura analógica)

bool isCurtainOpen = false; // Bandera para rastrear el estado de la cortina

void setup() {
  // Iniciar comunicación serial
  Serial.begin(115200);

  // Inicializar comunicación I2C para el LCD
  Wire.begin();

  // Inicializar LCD
  lcd.init(); // Inicializar el LCD
  lcd.backlight(); // Encender la retroiluminación del LCD

  // Inicializar sensor DHT
  dht.begin();

  // Inicializar motor paso a paso
  stepper.setMaxSpeed(2000); // Aumentar la velocidad máxima del motor
```

```
  stepper.setAcceleration(1000); // Aumentar la aceleración del motor

  // Configurar pin analógico del LDR como entrada
  pinMode(LDR_AO_PIN, INPUT);

  // Configurar pines del HC-SR04
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  // Mensaje de configuración completa
  Serial.println("Configuración completa.");
}

void loop() {
  // Leer temperatura y humedad
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // Mostrar temperatura y humedad en el LCD
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temperature);
  lcd.print(" C");
  lcd.setCursor(0, 1);
  lcd.print("Hum: ");
  lcd.print(humidity);
  lcd.print(" %");

  // Leer valor analógico del LDR
  int ldrAnalogValue = analogRead(LDR_AO_PIN);

  // Mostrar luminosidad en el LCD
  lcd.setCursor(0, 2);
  lcd.print("Luz: ");
  if (ldrAnalogValue > LIGHT_THRESHOLD) {
    lcd.print("Baja ");
  } else {
    lcd.print("Alta ");
  }

  // Controlar cortinas basado en el valor del LDR y las condiciones
ambientales
  if (ldrAnalogValue > LIGHT_THRESHOLD || temperature > TEMP_THRESHOLD ||
humidity > HUMIDITY_THRESHOLD) {
    abrirCortina();
  } else {
```

```
    cerrarCortina();
  }

  // Leer distancia del HC-SR04 para detectar cortina completamente
cerrada/abierta
  long duration, distance;
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  duration = pulseIn(ECHO_PIN, HIGH);
  distance = (duration / 2) / 29.1; // Convertir duración a distancia (cm)

  // Verificar si la cortina está completamente cerrada o abierta
  if (distance > 200 && !isCurtainOpen) { // se colocaron 200 cm a modo de
prueba
    // La cortina está completamente abierta
    isCurtainOpen = false;
    detenerCortina();
  } else if (distance < 40 && isCurtainOpen) { // se colocaron 40 cm a modo de
prueba
    // La cortina está completamente cerrada
    isCurtainOpen = true;
    detenerCortina();
  }

  // Retardo antes de la próxima iteración del bucle
  delay(500); // Ajustar retardo según necesidad
}

void abrirCortina() {
  if (!isCurtainOpen) {
    // Mostrar estado en el LCD
    lcd.setCursor(0, 3);
    lcd.print("Abriendo cortina    ");
    // Girar motor en sentido horario para abrir la cortina
    digitalWrite(DIR_PIN, HIGH);
    stepper.move(10000); // Mover motor paso a paso
    stepper.runSpeedToPosition(); // Ejecutar el movimiento
  }
}

void cerrarCortina() {
  if (isCurtainOpen) {
    // Mostrar estado en el LCD
    lcd.setCursor(0, 3);
```

```
    lcd.print("Cerrando cortina    ");
    // Girar motor en sentido antihorario para cerrar la cortina
    digitalWrite(DIR_PIN, LOW);
    stepper.move(-10000); // Mover motor paso a paso
    stepper.runSpeedToPosition(); // Ejecutar el movimiento
  }
}

void detenerCortina() {
  // Detener el motor
  stepper.stop();
  // Mostrar estado en el LCD
  lcd.setCursor(0, 3);
  lcd.print("Cortina detenida    ");
}
```