Logotipo

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.

**Unidad Académica Multidisciplinaria Mante.**

**Nombre de Practica:**

**Documentación Proyecto Sensores de Movimiento.**

**Alumnos:**

**Castelán Gamoneda Isaac**

**Calderón Maldonado Jahir Alejandro**

**Castro Marquez Cristian**

**López Camacho Jonathan Valentín**

**Ramírez Bustos Joshua Romany**

**Materia: Programación de Interfaces y Puertos USB.**

**Docente: Ing. Daniel López Piña.**

**Cd. Mante Marzo 2025**

**Definición del Proyecto.**

El proyecto consiste en el desarrollo de un sistema automatizado basado en Arduino UNO y el sensor ultrasónico HC-SR04 para detectar el movimiento de personas y medir la distancia a la que se encuentran. Al detectar movimiento, el sistema encenderá automáticamente los LED correspondientes a las áreas de puerta, sala y cocina. La interfaz de usuario, desarrollada en Python, permitirá al usuario activar o desactivar el sensor y controlar manualmente cada LED de manera independiente. Si el sensor de la cocina está desactivado, el LED correspondiente no se encenderá al detectar movimiento, lo que proporciona un control más preciso y eficiente del sistema.

**Objetivo Principal.**

Desarrollar un sistema basado en Arduino UNO y el sensor ultrasónico HC-SR04 para detectar el movimiento de personas, medir la distancia a la que pasan y activar automáticamente los LEDs de las áreas de puerta, sala y cocina. El sistema incluye una interfaz en Python para controlar manualmente cada LED y activar o desactivar el sensor de forma independiente.

**Alcance que Tendrá.**

* **Detección de movimiento** mediante el sensor HC-SR04.
* **Encendido automático de LEDs** en las áreas de puerta, sala y cocina al detectar movimiento.
* **Medición de distancia** desde el sensor hasta la persona.
* **Control manual** de cada LED mediante botones independientes en la interfaz en Python.
* **Activación y desactivación del sensor** desde la interfaz en Python.
* **Condicionalidad de activación**: Si el sensor de la cocina está desactivado, el LED correspondiente no se encenderá.

**Requisitos para el Proyecto.**

Requisitos de Hardware:

* Arduino UNO
* Sensor HC-SR04
* LED (para puerta, sala y cocina)
* Resistencias adecuadas
* Placa de pruebas
* Cables de conexión

Requisitos de Software:

* IDE de Arduino
* Python (con librerías de interfaz gráfica como tkinter)

**Fases del Proyecto.**

**Fase 1. Diseño.**

Esquema del Circuito y Conexiones.

Diagrama

El contenido generado por IA puede ser incorrecto.Diagrama

El contenido generado por IA puede ser incorrecto.

El arduino va conectado a la compu, los sensores van el vcc a 5 volts, el gnd a gnd, los leds el van en 2, 3, 4 y 5 el trid va en el pin 7 del arduino y del del sensor 2 va en el pin 6 y por último el echo del sensor 1 va al pin 9 y el del sensor 2 va al pin 10

Interfaz Gráfica.

Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.

**Fase 2. Implementación**

**Programación de Microcontrolador.**

Este código es un programa para Arduino que controla luces mediante sensores ultrasónicos y comandos recibidos por comunicación serial. El sistema usa cuatro sensores ultrasónicos (para puerta, poste, sala y cocina) conectados a pines específicos de trigger (TRIG) y eco (ECHO). Cada sensor tiene un LED asociado que se enciende cuando se detecta un objeto a menos de 40 cm (umbral definido en DISTANCE\_THRESHOLD), pero solo si el sensor correspondiente está activado (controlado por las variables booleanas puerta\_activo, poste\_activo, etc.).

En el loop(), el Arduino verifica constantemente si hay comandos enviados desde la PC (por ejemplo, "puerta:ON" para activar el sensor de la puerta o "luz\_poste:OFF" para apagar manualmente su LED). Además, si un sensor está activo y detecta un objeto cercano, enciende automáticamente su LED correspondiente (cambiando el estado de variables como luz\_puerta\_encendida). La función medirDistancia() calcula la distancia usando el tiempo de retardo del eco ultrasónico y la convierte a centímetros.

Finalmente, los LEDs se actualizan según sus estados (HIGH o LOW), permitiendo un control tanto automático (por proximidad) como manual (por comandos seriales). El pequeño delay(100) al final del loop() evita sobrecargar el procesador.

// Pines para los sensores ultrasónicos

const int TRIG\_PUERTA = 6;

const int ECHO\_PUERTA = 9;

const int TRIG\_POSTE = 7;

const int ECHO\_POSTE = 10;

const int TRIG\_SALA = 6;

const int ECHO\_SALA = 9;

const int TRIG\_COCINA = 6;

const int ECHO\_COCINA = 9;

// LEDs que se activan con los sensores

const int LED\_PUERTA = 3;

const int LED\_POSTE= 4;

const int LED\_SALA = 2;

const int LED\_COCINA = 5;

// Estados de activación de los sensores

bool puerta\_activo = true;

bool poste\_activo = true;

bool sala\_activo = true;

bool cocina\_activo = true;

// Estados de los LEDs (para mantener encendidos)

bool luz\_puerta\_encendida = false;

bool luz\_poste\_encendida = false;

bool luz\_sala\_encendida = false;

bool luz\_cocina\_encendida = false;

// Umbral de distancia para activar el LED (en cm)

const int DISTANCE\_THRESHOLD = 40;

void setup() {

Serial.begin(9600);

// Configurar sensores ultrasónicos

pinMode(TRIG\_PUERTA, OUTPUT);

pinMode(ECHO\_PUERTA, INPUT);

pinMode(TRIG\_POSTE, OUTPUT);

pinMode(ECHO\_POSTE, INPUT);

pinMode(TRIG\_SALA, OUTPUT);

pinMode(ECHO\_SALA, INPUT);

pinMode(TRIG\_COCINA , OUTPUT);

pinMode(ECHO\_COCINA , INPUT);

// Configurar LEDs

pinMode(LED\_PUERTA, OUTPUT);

pinMode(LED\_POSTE, OUTPUT);

pinMode(LED\_SALA, OUTPUT);

pinMode(LED\_COCINA, OUTPUT);

}

void loop() {

// Leer comandos de la PC

if (Serial.available()) {

String comando = Serial.readStringUntil('\n');

if (comando == "puerta:ON") puerta\_activo = true;

if (comando == "puerta:OFF") puerta\_activo = false;

if (comando == "poste:ON") poste\_activo = true;

if (comando == "poste:OFF") poste\_activo = false;

if (comando == "sala:ON") sala\_activo = true;

if (comando == "sala:OFF") sala\_activo = false;

if (comando == "cocina:ON") cocina\_activo = true;

if (comando == "cocina:OFF") cocina\_activo = false;

// También permitir apagar las luces manualmente

if (comando == "luz\_puerta:OFF") luz\_puerta\_encendida = false;

if (comando == "luz\_poste:OFF") luz\_poste\_encendida = false;

if (comando == "luz\_sala:OFF") luz\_sala\_encendida = false;

if (comando == "luz\_cocina:OFF") luz\_cocina\_encendida = false;

if (comando == "luz\_puerta:ON") luz\_puerta\_encendida = true;

if (comando == "luz\_poste:ON") luz\_poste\_encendida = true;

if (comando == "luz\_sala:ON") luz\_sala\_encendida = true;

if (comando == "luz\_cocina:ON") luz\_cocina\_encendida = true;

}

// Verificar sensores SOLO si están activos

if (puerta\_activo && medirDistancia(TRIG\_PUERTA, ECHO\_PUERTA) < DISTANCE\_THRESHOLD)

luz\_puerta\_encendida = true;

if (poste\_activo && medirDistancia(TRIG\_POSTE, ECHO\_POSTE) < DISTANCE\_THRESHOLD)

luz\_poste\_encendida = true;

if (sala\_activo && medirDistancia(TRIG\_SALA, ECHO\_SALA) < DISTANCE\_THRESHOLD)

luz\_sala\_encendida = true;

if (cocina\_activo && medirDistancia(TRIG\_COCINA, ECHO\_COCINA) < DISTANCE\_THRESHOLD)

luz\_cocina\_encendida = true;

// Control de LEDs basado en los estados de encendido

digitalWrite(LED\_PUERTA, luz\_puerta\_encendida ? HIGH : LOW);

digitalWrite(LED\_POSTE, luz\_poste\_encendida ? HIGH : LOW);

digitalWrite(LED\_SALA, luz\_sala\_encendida ? HIGH : LOW);

digitalWrite(LED\_COCINA, luz\_cocina\_encendida ? HIGH : LOW);

delay(100); // Pequeño retardo para estabilidad

}

// Función para medir la distancia con el sensor ultrasónico

float medirDistancia(int trig, int echo) {

digitalWrite(trig, LOW);

delayMicroseconds(2);

digitalWrite(trig, HIGH);

delayMicroseconds(10);

digitalWrite(trig, LOW);

long duration = pulseIn(echo, HIGH);

float distancia = duration \* 0.017; // Convertir tiempo a cm

Serial.print("Distancia: ");

Serial.print(distancia);

Serial.println(" cm");

return distancia;

}

**Programación de Interfaz.**

Este código crea una interfaz gráfica con Tkinter para controlar sensores y luces conectados a un Arduino. La clase ArduinoController maneja la conexión serial con el Arduino (o simula la conexión si no se detecta) y envía comandos para activar/desactivar sensores y encender/apagar luces. La función crear\_interfaz construye una interfaz moderna con pestañas: una para controlar 4 sensores (puerta, poste, sala, cocina) y otra para 4 luces, con botones de encendido/apagado que cambian de color según el estado. La interfaz incluye un panel de estado y un botón para actualizar. Los comandos se envían en formato "tipo\_dispositivo:acción" (ej: "luz\_puerta:ON") y la aplicación actualiza los estados internos y la interfaz visualmente.

import tkinter as tk

import serial

from tkinter import font as tkFont

import serial.tools.list\_ports

from tkinter import ttk

class ArduinoController:

def \_init\_(self):

self.arduino = self.conectar\_arduino()

self.sensor\_states = {

"puerta": True,

"poste": True,

"sala": True,

"cocina": True

}

self.light\_states = {

"puerta": False,

"poste": False,

"sala": False,

"cocina": False

}

def conectar\_arduino(self):

try:

puertos = serial.tools.list\_ports.comports()

for puerto in puertos:

if "Arduino" in puerto.description or "CH340" in puerto.description:

arduino = serial.Serial(puerto.device, 9600, timeout=1)

print(f"✨ Conectado a {puerto.device}")

return arduino

print("⚠ Modo simulación (Arduino no detectado)")

return None

except Exception as e:

print(f"🔥 Error de conexión: {str(e)}")

return None

def enviar\_comando(self, tipo, dispositivo, accion):

comando = f"{tipo}\_{dispositivo}:{accion}\n" if tipo == "luz" else f"{dispositivo}:{accion}\n"

# Actualizar estados

if tipo == "sensor":

self.sensor\_states[dispositivo] = (accion == "ON")

else:

self.light\_states[dispositivo] = (accion == "ON")

if self.arduino:

try:

self.arduino.write(comando.encode())

print(f"📤 Enviado: {comando.strip()}")

except Exception as e:

print(f"⚠ Error al enviar: {str(e)}")

self.arduino = self.conectar\_arduino()

else:

print(f"🖥 Simulación: {comando.strip()}")

def crear\_interfaz(controller):

root = tk.Tk()

root.title("🌟 Control de Sensores Premium")

root.geometry("800x600")

root.configure(bg="#2c3e50")

# Estilo moderno

style = ttk.Style()

style.theme\_use('clam')

style.configure('TFrame', background='#2c3e50')

style.configure('TLabel', background='#2c3e50', foreground='#ecf0f1', font=('Helvetica', 12))

style.configure('Title.TLabel', font=('Helvetica', 16, 'bold'))

style.configure('Sensor.TButton', font=('Helvetica', 10), padding=10)

style.configure('Light.TButton', font=('Helvetica', 10), padding=8)

style.map('On.TButton', background=[('active', '#27ae60'), ('!active', '#2ecc71')])

style.map('Off.TButton', background=[('active', '#c0392b'), ('!active', '#e74c3c')])

# Marco principal con efecto de tarjeta

main\_frame = ttk.Frame(root, style='TFrame', padding=(20, 20))

main\_frame.pack(expand=True, fill="both")

# Título con icono

ttk.Label(main\_frame, text="⚡ Control de Sensores Inteligente", style='Title.TLabel').pack(pady=(0, 20))

# Sección de Sensores (con pestañas)

notebook = ttk.Notebook(main\_frame)

notebook.pack(fill="both", expand=True, padx=10, pady=10)

# Pestaña de Sensores

sensor\_tab = ttk.Frame(notebook)

notebook.add(sensor\_tab, text="🧭 Control de Sensores")

# Grid para sensores

for i, (sensor, estado) in enumerate(controller.sensor\_states.items()):

frame = ttk.Frame(sensor\_tab, style='TFrame')

frame.grid(row=i, column=0, pady=10, padx=20, sticky="ew")

icon = "🚪" if sensor == "puerta" else "💡" if sensor == "poste" else "🛋" if sensor == "sala" else "🍳"

ttk.Label(frame, text=f"{icon} Sensor {sensor.capitalize()}", style='TLabel').pack(side="left")

btn\_frame = ttk.Frame(frame, style='TFrame')

btn\_frame.pack(side="right")

btn\_on = ttk.Button(btn\_frame, text="ACTIVAR", style='On.TButton',

command=lambda s=sensor: controller.enviar\_comando("sensor", s, "ON"))

btn\_on.pack(side="left", padx=5)

btn\_off = ttk.Button(btn\_frame, text="DESACTIVAR", style='Off.TButton',

command=lambda s=sensor: controller.enviar\_comando("sensor", s, "OFF"))

btn\_off.pack(side="left")

# Estado inicial

if estado:

btn\_on.state(['pressed', 'disabled'])

else:

btn\_off.state(['pressed', 'disabled'])

# Pestaña de Luces

light\_tab = ttk.Frame(notebook)

notebook.add(light\_tab, text="💡 Control de Luces")

# Grid para luces

light\_icons = {"puerta": "🚪", "poste": "🪔", "sala": "🛋", "cocina": "🍳"}

for i, (luz, estado) in enumerate(controller.light\_states.items()):

frame = ttk.Frame(light\_tab, style='TFrame')

frame.grid(row=i//2, column=i%2, pady=15, padx=20, sticky="nsew")

icon = light\_icons[luz]

ttk.Label(frame, text=f"{icon} Luz {luz.capitalize()}", style='TLabel').pack()

btn\_frame = ttk.Frame(frame, style='TFrame')

btn\_frame.pack(pady=5)

btn\_on = ttk.Button(btn\_frame, text="ENCENDER", style='On.TButton',

command=lambda l=luz: controller.enviar\_comando("luz", l, "ON"))

btn\_on.pack(side="left", padx=5)

btn\_off = ttk.Button(btn\_frame, text="APAGAR", style='Off.TButton',

command=lambda l=luz: controller.enviar\_comando("luz", l, "OFF"))

btn\_off.pack(side="left")

# Estado inicial

if estado:

btn\_on.state(['pressed', 'disabled'])

else:

btn\_off.state(['pressed', 'disabled'])

# Panel de estado

status\_frame = ttk.Frame(main\_frame, style='TFrame')

status\_frame.pack(fill="x", pady=(15, 0))

ttk.Label(status\_frame, text="🔄 Estado: ", style='TLabel').pack(side="left")

connection\_status = ttk.Label(status\_frame,

text="Conectado" if controller.arduino else "Modo simulación",

style='TLabel')

connection\_status.pack(side="left")

# Efectos hover

def on\_enter(e):

e.widget['background'] = '#3498db'

def on\_leave(e):

e.widget['background'] = '#2980b9'

# Botón de actualización con efecto

refresh\_btn = tk.Button(status\_frame, text="🔄 Actualizar", bg="#2980b9", fg="white",

relief="flat", bd=0, padx=15, pady=5,

command=lambda: actualizar\_estados(controller, root))

refresh\_btn.pack(side="right")

refresh\_btn.bind("<Enter>", on\_enter)

refresh\_btn.bind("<Leave>", on\_leave)

return root

def actualizar\_estados(controller, window):

print("♻ Actualizando estados...")

window.destroy()

crear\_interfaz(controller).mainloop()

if \_name\_ == "\_main\_":

print("🚀 Iniciando interfaz de control...")

controller = ArduinoController()

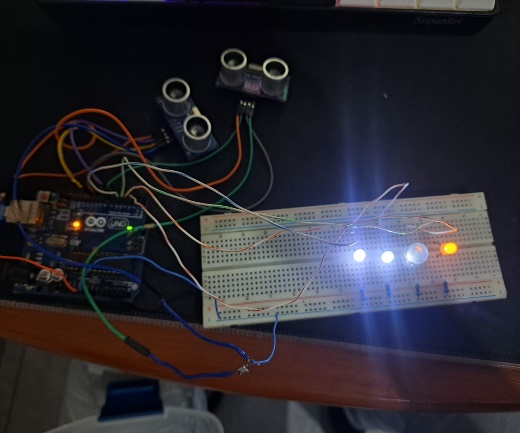
app = crear\_interfaz(controller)

app.mainloop()

**Fase 3. Pruebas de Funcionamiento.**

En las siguientes imágenes se ilustra el funcionamiento del sistema desarrollado. El sensor ultrasónico es capaz de detectar la presencia o el movimiento de un objeto dentro de su rango operativo mediante la emisión y recepción de señales ultrasónicas. Cuando el sensor detecta un objeto o movimiento dentro de la distancia, envía una señal al microcontrolador Arduino UNO, el cual procesa la información y determina la activación correspondiente de los LEDs asociados. La activación de los LED es gestionada automáticamente por el microcontrolador y puede ser anulada manualmente mediante la interfaz gráfica, que permite encender o apagar cada LED de manera independiente.

**Un ratón de computadora

El contenido generado por IA puede ser incorrecto.**

**Fase 4. Montaje**

En las siguientes imágenes se puede apreciar el avance de como fue el montaje en donde ira el circuito el cual será una simulación ambientada en el hogar teniendo como pruebas las partes de la cocina, sala, puerta de entrada y un poste.

Texto

El contenido generado por IA puede ser incorrecto.Imagen que contiene verde, tabla, caja, hombre

El contenido generado por IA puede ser incorrecto.Imagen que contiene interior, verde, computadora, tabla

El contenido generado por IA puede ser incorrecto.