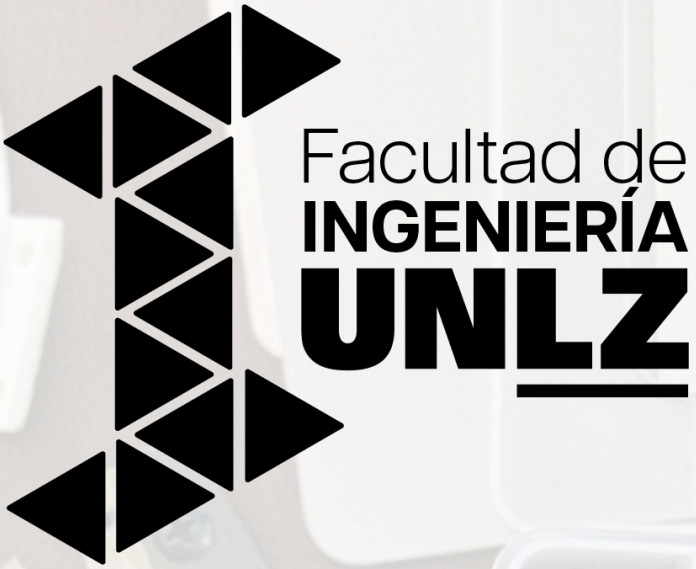
Proyecto tecnológico 2



Modelo de Casa Domótica Inteligente con IoT y Reconocimiento de Voz

Ing. Lukaszewicz, Cristian

Documento Versión 1.0

Índice

1. Observación del Entorno

1.1. Disparador

En la actualidad, la automatización del hogar se ha convertido en una tendencia creciente, facilitando la vida diaria mediante el control de dispositivos electrónicos y la monitorización del entorno. Nos preguntamos: ¿Cómo podemos usar comandos de voz e IoT para controlar diferentes aspectos de una casa inteligente a través de un modelo a escala?

1.2. Recorte del Tema

Desarrollar un modelo a escala de una casa domótica que permita a los estudiantes controlar luces, ventilación y monitorear variables como temperatura y humedad mediante comandos de voz y una plataforma IoT en línea. Utilizaremos tecnologías como Whisper para el reconocimiento de voz local y sensores IoT para la monitorización en tiempo real.

1.3. Objetivos

1.3.1. Objetivo General

Crear un modelo funcional de una casa domótica inteligente que integre tecnologías de IoT y reconocimiento de voz local, permitiendo el control y monitoreo de diferentes aspectos del hogar mediante comandos de voz y una plataforma en línea.

1.3.2. Objetivos Específicos

- Implementar el reconocimiento de voz local para controlar dispositivos electrónicos.
- Integrar sensores IoT para monitorear variables ambientales.
- Desarrollar una plataforma en línea para visualizar datos en tiempo real.
- Fomentar el trabajo en equipo y el desarrollo de competencias técnicas y blandas en los estudiantes.

1.4. Materiales para Ejecutar el Proyecto

- **Hardware:**
- Raspberry Pi o Arduino con módulos Wi-Fi
- Netbook con Windows y conexión a Arduino (en caso de no disponer de Raspberry Pi)

- Micrófono USB
- Relés para controlar luces y ventiladores
- Sensores de temperatura y humedad (DHT11 o DHT22)
- LEDs y ventiladores a escala
- Cables y componentes electrónicos adicionales
- Cámara USB (para reconocimiento facial opcional)
- **Software:**
- Plataforma IoT (ThingSpeak, Blynk, etc.)
- Software de programación (Python)
- Bibliotecas de reconocimiento de voz (Whisper local, Transformers para clasificación de texto)
- IDE de Arduino
- **Otros:**
- Acceso a internet
- Protoboard y resistencias
- Fuentes de alimentación
- Caja de distribución para el montaje del modelo a escala
- Herramientas básicas de electrónica (soldador, multímetro, etc.)

1.5. Otros Actores Involucrados

- Docentes de programación, electrónica e inteligencia artificial
- Asistentes técnicos para la configuración de plataformas IoT
- Coordinadores de proyectos tecnológicos
- Proveedores de hardware y software
- Tutores externos (opcional) para soporte adicional

1.6. Roles y Funciones a Designar en Cada Grupo de Trabajo

1.6.1. Equipos

Cada grupo se dividirá en:

- **Líder de Equipo:** Coordina actividades y mantiene el progreso.
- **Programadores:** Configuran los comandos de voz y la programación en Python.
- **Encargados del Hardware:** Montan sensores y dispositivos de control.
- **Testers:** Realizan pruebas del sistema.

- **Documentalistas:** Se encargan de la documentación y reportes del proyecto.

1.7. Tiempo Total del Proyecto

El proyecto se desarrollará en cuatro semanas, cada una enfocada en diferentes aspectos del desarrollo e integración del sistema domótico.

1.8. Plan de Trabajo

1.8.1. Planteamiento Guía

"En una casa a escala, queremos implementar un sistema domótico que permita controlar luces, ventiladores y monitorear el ambiente mediante comandos de voz y una plataforma IoT. ¿Qué tecnologías necesitamos y cómo las integraremos para crear un sistema eficiente y funcional?"

1.8.2. Organización y Planificación

El proyecto se divide en cuatro etapas semanales, cada una con tareas específicas de investigación, desarrollo e integración.

1.8.3. Descripción Detallada de Actividades

2. Diagnóstico

2.1. Conocimiento Técnico

Para llevar a cabo este proyecto, es esencial que los estudiantes tengan conocimientos básicos en:

- **Programación en Python:** Para el desarrollo de scripts que manejan el reconocimiento de voz y la integración con IoT.
- **Manejo de Raspberry Pi o Arduino:** Configuración y programación de estos dispositivos para controlar hardware.
- **Uso de Sensores IoT:** Instalación y calibración de sensores para monitoreo ambiental.
- **Bibliotecas de Reconocimiento de Voz:** Uso de Whisper local o Transformers para clasificación de texto para implementar comandos de voz.

2.2. Aporte Curricular

Este proyecto contribuye al currículo en las siguientes áreas:

- **Programación:** Aplicación práctica de conceptos de programación en Python.
- **Inteligencia Artificial:** Implementación de reconocimiento de voz mediante tecnologías de IA.
- **Electrónica:** Integración y control de sensores y dispositivos electrónicos.
- **IoT:** Configuración y uso de plataformas IoT para el monitoreo y control de dispositivos.

2.3. Aporte al Desarrollo de Competencias Genéricas

- **Resolución de Problemas:** Identificación y solución de desafíos técnicos durante el desarrollo del proyecto.
- **Trabajo en Equipo:** Colaboración efectiva entre los miembros del grupo.
- **Integración de Tecnologías:** Uso combinado de IA e IoT para crear un sistema funcional.
- **Planificación de Proyectos:** Organización y gestión del tiempo y recursos para cumplir con los objetivos del proyecto.
- **Organización:** Coordinación de tareas y roles dentro del equipo.

3. Planteo de las Metas

3.1. Selección de Contenidos

- **Reconocimiento de Voz:** Uso de Whisper local para implementar comandos de voz.
- **Sensores IoT:** Instalación y uso de sensores de temperatura y humedad.
- **Control de Dispositivos:** Implementación de relés para controlar luces y ventiladores.
- **Plataforma de Monitoreo:** Configuración de plataformas como ThingSpeak o Blynk para la visualización de datos en tiempo real.

3.2. Producto Final

Un modelo a escala de una casa inteligente controlada mediante comandos de voz y monitoreada en tiempo real a través de una plataforma IoT, permitiendo el control de luces, ventiladores y la monitorización de variables ambientales.

3.3. Objetivos de Aprendizaje

- **Comprender la Domótica:** Integración de tecnologías como IA e IoT en sistemas de automatización del hogar.
- **Aplicar Programación y Electrónica:** Desarrollo de scripts en Python y manejo de hardware para crear un sistema interactivo.
- **Desarrollar Habilidades Técnicas y Blandas:** Trabajo en equipo, planificación y resolución de problemas técnicos.

3.4. Resultados de Aprendizaje

- **Sistema Domótico Funcional:** Desarrollo de un sistema que permite el control por voz y la monitorización de datos en tiempo real.
- **Interacción con Dispositivos Inteligentes:** Capacidad de interactuar con luces y ventiladores mediante comandos de voz.
- **Monitoreo Ambiental:** Visualización de datos de temperatura y humedad en una plataforma IoT.

4. Acciones a Desarrollar

4.1. Semana 1: Introducción a Domótica e IoT

Objetivo de la Etapa:
Introducir a los estudiantes en los conceptos básicos de domótica e IoT, familiarizándolos con los sensores y dispositivos que se utilizarán en el proyecto.

Actividades:

- **Conceptos Básicos de Domótica:**
- **Definición:** La domótica se refiere al uso de tecnologías avanzadas para automatizar y controlar diversos aspectos del hogar, como la iluminación, la climatización, la seguridad y otros sistemas electrónicos.
- **Aplicaciones:** Automatización de luces, sistemas de seguridad, control de temperatura, gestión de energía, etc.
- **Introducción a Sensores de Temperatura y Humedad:**
- **Funcionamiento:** Explicación de cómo funcionan los sensores DHT11 y DHT22, sus diferencias y aplicaciones.
- **Tipos:** Sensores digitales vs. analógicos.
- **Control de Dispositivos Electrónicos:**
- **Relés:** Explicación sobre qué son los relés, cómo funcionan y su uso en el control de dispositivos de alto voltaje mediante microcontroladores.
- **Conexión:** Ejemplos de cómo conectar un relé a una Raspberry Pi o Arduino.
- **Configuración de Raspberry Pi o Arduino:**
- **Raspberry Pi:**
- **Instalación del Sistema Operativo:** Descargar e instalar Raspberry Pi OS utilizando herramientas como Raspberry Pi Imager.
- **Configuración Inicial:** Configuración de red, actualización del sistema.
- **Instalación de Python y Bibliotecas Necesarias:**

```
sudo apt update
sudo apt upgrade -y
sudo apt install python3-pip -y
```

```
pip3 install openai-whisper
pip3 install Adafruit_DHT
pip3 install requests
pip3 install sounddevice soundfile
```

- **Arduino con Netbook (Windows):**
- **Instalación del IDE de Arduino:** Descargar e instalar el IDE desde Arduino.
- **Configuración del Entorno:** Seleccionar el modelo de Arduino y el puerto correspondiente en el IDE.
- **Instalación de Bibliotecas:** Instalar bibliotecas necesarias para sensores y comunicación.
- En el IDE, ir a **Sketch > Include Library > Manage Libraries...** y buscar e instalar:
- `DHT sensor library`
- `Adafruit Unified Sensor`

Alternativa con Netbook y Arduino en Windows:

- **Configuración del Entorno de Desarrollo:**
- **Instalar Python en la Netbook:**
- Descargar e instalar Python desde python.org.
- Durante la instalación, seleccionar la opción "Add Python to PATH".
- **Instalar Pip y PySerial:**
- Abrir la línea de comandos (CMD) y ejecutar:

```
python -m pip install --upgrade pip

pip install pyserial sounddevice soundfile
transformers
```

- **Configurar Comunicación entre Netbook y Arduino:**
- **Conectar el Arduino a la Netbook:** Utilizar un cable USB para conectar el Arduino a la netbook.

- **Verificar el Puerto COM:** En el Administrador de Dispositivos de Windows, identificar el puerto COM asignado al Arduino.
- **Script de Python para recibir comandos de voz y enviar instrucciones al Arduino:**

```
import whisper

import serial

import time

import sounddevice as sd

import soundfile as sf

from transformers import pipeline


# Configuración de Whisper

model = whisper.load_model("base")


# Configuración de Arduino (ajustar el puerto COM
según corresponda)

ser = serial.Serial('COM3', 9600, timeout=1)


# Cargar modelo de clasificación de texto

classifier = pipeline("zero-shot-classification",
model="facebook/bart-large-mnli")

candidate_labels = ["encender luces", "apagar
luces", "encender ventilador", "apagar ventilador"]
```

```
def capturar_audio(duracion=5, fs=44100):
    print("Capturando audio...")
    audio = sd.rec(int(duracion * fs),
samplerate=fs, channels=1)
    sd.wait()
    sf.write('captura.wav', audio, fs)
    return 'captura.wav'

def procesar_comando(audio_path):
    # Transcribir audio con Whisper
    result = model.transcribe(audio_path)
    texto = result["text"].lower()
    print(f"Comando detectado: {texto}")

    # Clasificar el texto para identificar el
comando
    classification = classifier(texto,
candidate_labels)
    label = classification['labels'][0]

    if label == "encender luces":
```

```
        ser.write(b'ENCENDER_LUZ\n')
        print("Luces encendidas")
    elif label == "apagar luces":
        ser.write(b'APAGAR_LUZ\n')
        print("Luces apagadas")
    elif label == "encender ventilador":
        ser.write(b'ENCENDER_VENTILADOR\n')
        print("Ventilador encendido")
    elif label == "apagar ventilador":
        ser.write(b'APAGAR_VENTILADOR\n')
        print("Ventilador apagado")
    else:
        print("Comando no reconocido")

# Loop principal
while True:
    audio = capturar_audio()
    procesar_comando(audio)
    time.sleep(1)
```

- **Notas:**
- **Bibliotecas Necesarias:** `whisper`, `pyserial`, `sounddevice`, `soundfile`, `transformers`.

- **Función** `capturar_audio`**:** Captura un fragmento de audio de 5 segundos y lo guarda como `captura.wav`.
- **Función** `procesar_comando`**:** Transcribe el audio utilizando Whisper, clasifica el texto con un modelo de clasificación de texto y envía comandos al Arduino mediante serial.
- **Modelo de Clasificación:** Utiliza `facebook/bart-large-mnli` para clasificación de texto sin necesidad de APIs de pago.

Dinámica de Introducción:

- **Preguntas de Guía:**
- ¿Qué es una casa domótica?
- ¿Cómo puede la tecnología mejorar la eficiencia en el hogar?
- ¿Qué dispositivos domóticos conoces?
- **Discusión en Grupo:**
- Intercambio de ideas sobre las ventajas y desafíos de implementar una casa inteligente.
- Reflexión sobre cómo la domótica puede contribuir a la sostenibilidad y eficiencia energética.

Investigación Inicial:

- **Exploración de Componentes:**
- Identificación y descripción de los sensores y dispositivos que se utilizarán en el proyecto.
- Revisión de datasheets y manuales de los sensores DHT11/DHT22, relés, y otros componentes.
- **Revisión de Proyectos Similares:**
- Análisis de casos de éxito en domótica y proyectos de IoT en el hogar.
- Identificación de buenas prácticas y posibles mejoras.

Semana 2: Reconocimiento de Voz y Control de Dispositivos

Objetivo de la Etapa:
Implementar el reconocimiento de voz local utilizando Whisper y desarrollar comandos básicos para el control de dispositivos electrónicos.

Actividades:

Implementación de Whisper Local

En Raspberry Pi:

- **Instalación de Whisper:**

```
pip install
git+https://github.com/openai/whisper.git
```

- **Prueba Básica de Whisper:**

```
import whisper


model = whisper.load_model("base")
result = model.transcribe("audio_sample.wav")
print(result["text"])
```

En Netbook con Windows:

- **Instalación de Whisper:**
- **Instalar Git:** Descargar e instalar Git desde git-scm.com.
- **Instalar Whisper:**
- Abrir la línea de comandos (CMD) y ejecutar:

```
pip install
```

```
git+https://github.com/openai/whisper.git
```

- **Prueba Básica de Whisper:**

```
import whisper


model = whisper.load_model("base")
result = model.transcribe("audio_sample.wav")
print(result["text"])
```

Configuración del Micrófono:

- **Verificar Conexión:**
- Asegurarse de que el micrófono USB esté correctamente conectado y reconocido por el sistema.
- **Probar la Captura de Audio:**
- **En Raspberry Pi:**

```
arecord -D plughw:1,0 -d 5 test.wav
aplay test.wav
```

- **En Netbook con Windows:**
- Usar aplicaciones como **Grabadora de Voz** para grabar y reproducir un archivo de audio de prueba.

Desarrollo de Comandos de Voz

Definición de Comandos:

- "Encender luces"
- "Apagar ventilador"

- "Aumentar temperatura a X grados"
- "Disminuir temperatura a X grados"

Programación en Python:

Código Unificado para Raspberry Pi y Netbook con Windows:

```
import whisper
import time
import sounddevice as sd
import soundfile as sf

# Importar según el dispositivo
import platform
if platform.system() == "Linux":
    import RPi.GPIO as GPIO
elif platform.system() == "Windows":
    import serial
    from transformers import pipeline

# Configuración específica para cada plataforma
if platform.system() == "Linux":
    # Configuración de GPIO para Raspberry Pi
    GPIO.setmode(GPIO.BCM)
```

```
    GPIO.setup(18, GPIO.OUT)  # Luz
    GPIO.setup(23, GPIO.OUT)  # Ventilador

    # Cargar modelo Whisper
    model = whisper.load_model("base")

elif platform.system() == "Windows":
    # Configuración de Arduino (ajustar el puerto COM según corresponda)
    ser = serial.Serial('COM3', 9600, timeout=1)

    # Cargar modelo Whisper
    model = whisper.load_model("base")

    # Cargar modelo de clasificación de texto
    classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")
    candidate_labels = ["encender luces", "apagar luces", "encender ventilador", "apagar ventilador"]

def capturar_audio(duracion=5, fs=44100):
    print("Capturando audio...")
```

```
    audio = sd.rec(int(duracion * fs),
samplerate=fs, channels=1)
    sd.wait()
    sf.write('captura.wav', audio, fs)
    return 'captura.wav'

def procesar_comando(audio_path):
    # Transcribir audio con Whisper
    result = model.transcribe(audio_path)
    texto = result["text"].lower()
    print(f"Comando detectado: {texto}")

    if platform.system() == "Linux":
        # Procesar comandos en Raspberry Pi
        if "encender luces" in texto:
            GPIO.output(18, GPIO.HIGH)
            print("Luces encendidas")
        elif "apagar luces" in texto:
            GPIO.output(18, GPIO.LOW)
            print("Luces apagadas")
        elif "encender ventilador" in texto:
            GPIO.output(23, GPIO.HIGH)
```

```
            print("Ventilador encendido")
        elif "apagar ventilador" in texto:
            GPIO.output(23, GPIO.LOW)
            print("Ventilador apagado")
        elif "aumentar temperatura a" in texto:
            # Implementar lógica para aumentar
temperatura
            print("Temperatura aumentada")
        elif "disminuir temperatura a" in texto:
            # Implementar lógica para disminuir
temperatura
            print("Temperatura disminuida")


    elif platform.system() == "Windows":
        # Clasificar el texto para identificar el
comando
        classification = classifier(texto,
candidate_labels)
        label = classification['labels'][0]


        if label == "encender luces":
            ser.write(b'ENCENDER_LUZ\n')
            print("Luces encendidas")
```

```
        elif label == "apagar luces":
            ser.write(b'APAGAR_LUZ\n')
            print("Luces apagadas")
        elif label == "encender ventilador":
            ser.write(b'ENCENDER_VENTILADOR\n')
            print("Ventilador encendido")
        elif label == "apagar ventilador":
            ser.write(b'APAGAR_VENTILADOR\n')
            print("Ventilador apagado")
        else:
            print("Comando no reconocido")

while True:
    audio = capturar_audio()
    procesar_comando(audio)
    time.sleep(1)
```

Explicaciones Detalladas:

1. **Importaciones y Configuraciones Iniciales:**
    - **Importar Bibliotecas Comunes:**
    - `whisper`: Para el reconocimiento de voz local.
    - `time`, `sounddevice`, `soundfile`: Para la captura y manejo de audio.
    - **Importar Bibliotecas Específicas según la Plataforma:**
    - **Raspberry Pi (Linux):** `RPi.GPIO` para el control de pines GPIO.

- **Windows:** `serial` para la comunicación con Arduino y `transformers` para la clasificación de texto.

2. **Configuración Específica por Plataforma:**
   - **Raspberry Pi:**
   - Configura los pines GPIO para controlar los relés que manejarán las luces y el ventilador.
   - Carga el modelo Whisper para el reconocimiento de voz local.
   - **Windows (Netbook con Arduino):**
   - Establece la comunicación serial con el Arduino a través del puerto COM especificado.
   - Carga el modelo Whisper y el modelo de clasificación de texto (`facebook/bart-large-mnli`) para interpretar comandos de voz sin necesidad de APIs de pago.

3. **Funciones Principales:**
   - `capturar_audio`**:** Captura un fragmento de audio de 5 segundos y lo guarda como `captura.wav`.
   - `procesar_comando`**:**
   - **Transcripción:** Utiliza Whisper para transcribir el audio capturado a texto.
   - **Clasificación de Comandos (Windows):** Usa un modelo de clasificación de texto para identificar el comando específico.
   - **Ejecución de Comandos:**
   - **Raspberry Pi:** Controla los pines GPIO para encender o apagar luces y ventiladores según el texto transcrito.
   - **Windows:** Envía comandos seriales al Arduino para controlar los dispositivos.

4. **Loop Principal:**
   - Captura audio.
   - Procesa el comando detectado.
   - Espera 1 segundo antes de repetir el proceso.

Notas Importantes:

- **Puerto COM en Windows:**

- Asegúrate de reemplazar `'COM3'` con el puerto COM correcto asignado a tu Arduino. Puedes verificarlo en el Administrador de Dispositivos de Windows bajo "Puertos (COM & LPT)".
- **Manejo de Excepciones:**
- Para una mayor robustez, considera agregar bloques `try-except` para manejar posibles errores, como la falta de reconocimiento de voz o problemas de conexión serial.
- **Modelo de Clasificación:**
- El modelo `facebook/bart-large-mnli` es gratuito y puede ejecutarse localmente, evitando costos asociados a APIs de pago. Este modelo realiza clasificación de texto sin requerir datos adicionales de entrenamiento.

Control de Relés

Conexión de Relés a Raspberry Pi o Arduino:

- **Raspberry Pi:**
- **Esquema de Conexión:**
- **Relé 1 (Luces):**
- VCC del relé a 5V de Raspberry Pi.
- GND del relé a GND de Raspberry Pi.
- IN del relé a GPIO 18.
- Conectar la carga (LED) al relé según el esquema del fabricante.
- **Relé 2 (Ventilador):**
- VCC del relé a 5V de Raspberry Pi.
- GND del relé a GND de Raspberry Pi.
- IN del relé a GPIO 23.
- Conectar la carga (ventilador a escala) al relé.
- **Arduino con Netbook (Windows):**
- **Esquema de Conexión:**
- **Relé 1 (Luces):**
- VCC del relé a 5V de Arduino.
- GND del relé a GND de Arduino.
- IN del relé a pin digital 8 de Arduino.
- Conectar la carga (LED) al relé.
- **Relé 2 (Ventilador):**

- VCC del relé a 5V de Arduino.
- GND del relé a GND de Arduino.
- IN del relé a pin digital 9 de Arduino.
- Conectar la carga (ventilador a escala) al relé.

Código de Control de Relés:

El control de relés está integrado en el script unificado presentado anteriormente. Dependiendo de la plataforma, los comandos de voz detectados activarán los relés correspondientes a través de GPIO (Raspberry Pi) o mediante comunicación serial (Arduino).

Pruebas Iniciales:

- **Encender y Apagar Luces:**
- Pronunciar "Encender luces" y verificar que las luces se encienden.
- Pronunciar "Apagar luces" y verificar que las luces se apagan.
- **Encender y Apagar Ventilador:**
- Pronunciar "Encender ventilador" y verificar que el ventilador se enciende.
- Pronunciar "Apagar ventilador" y verificar que el ventilador se apaga.

Problemas y Soluciones:

- **Interferencias en el reconocimiento de voz:**
- Solución: Ajustar la sensibilidad del micrófono y utilizar filtros de ruido.
- **Comandos no reconocidos correctamente:**
- Solución: Asegurar una buena calidad de audio y ajustar el modelo de clasificación si es necesario.
- **Problemas de conexión serial en Windows:**
- Solución: Verificar que el puerto COM es el correcto y que no hay otros dispositivos usando el mismo puerto.

Semana 3: Integración de Sensores y Monitoreo con IoT

Objetivo de la Etapa:

Integrar sensores de temperatura y humedad con una plataforma IoT para el monitoreo en tiempo real y la visualización de datos.

Actividades:

Configuración de Sensores DHT11 o DHT22

- **Conexión Física:**
- **Raspberry Pi:**
- **DHT11/DHT22:**
- VCC a 5V de Raspberry Pi.
- GND a GND de Raspberry Pi.
- DATA a GPIO 4.
- Resistencia pull-up de 10kΩ entre VCC y DATA.
- **Arduino:**
- **DHT11/DHT22:**
- VCC a 5V de Arduino.
- GND a GND de Arduino.
- DATA a pin digital 2.
- Resistencia pull-up de 10kΩ entre VCC y DATA.
- **Calibración:**
- Verificar las lecturas iniciales utilizando scripts básicos.
- Ajustar la ubicación de los sensores para evitar interferencias de fuentes de calor/frío directas.

Integración con Plataforma IoT

ThingSpeak:

- **Crear una Cuenta y Configurar un Canal:**
- Registrarse en ThingSpeak y crear un nuevo canal.
- Configurar campos para temperatura y humedad.
- Obtener la API Key del canal para enviar datos.
- **Configuración del Script en Python:**

```
import Adafruit_DHT
```

```
import requests
import time


SENSOR = Adafruit_DHT.DHT22
PIN = 4
API_KEY = "TU_API_KEY"


def leer_sensor():
    humedad, temperatura =
Adafruit_DHT.read_retry(SENSOR, PIN)
    return humedad, temperatura


def enviar_a_thingspeak(humedad, temperatura):
    url = "https://api.thingspeak.com/update"
    params = {
        "api_key": API_KEY,
        "field1": temperatura,
        "field2": humedad
    }
    response = requests.get(url, params=params)
    if response.status_code == 200:
        print("Datos enviados correctamente")
```

```
    else:
        print("Error al enviar datos")


while True:
    humedad, temperatura = leer_sensor()
    if humedad is not None and temperatura is not
None:
        enviar_a_thingspeak(humedad, temperatura)
    else:
        print("Fallo al leer el sensor")
    time.sleep(60)  # Enviar datos cada 60 segundos
```

Blynk:

- **Crear un Proyecto en Blynk:**
- Descargar la aplicación Blynk en un dispositivo móvil.
- Crear un nuevo proyecto y obtener el Auth Token.
- Configurar widgets para recibir datos de temperatura y humedad.
- **Configuración del Script en Python:**

```
import Adafruit_DHT
import BlynkLib
import time


# Configuración del sensor
```

```
SENSOR = Adafruit_DHT.DHT22
PIN = 4


# Configuración de Blynk
BLYNK_AUTH = 'TU_AUTH_TOKEN'
blynk = BlynkLib.Blynk(BLYNK_AUTH)


# Vínculos de los widgets
TEMPERATURA = 1
HUMEDAD = 2


def leer_sensor():
    humedad, temperatura =
Adafruit_DHT.read_retry(SENSOR, PIN)
    return humedad, temperatura


while True:
    humedad, temperatura = leer_sensor()
    if humedad is not None and temperatura is not
None:
        blynk.virtual_write(TEMPERATURA,
temperatura)
```

```
        blynk.virtual_write(HUMEDAD, humedad)
        print(f"Temperatura: {temperatura} °C,
Humedad: {humedad} %")
    else:
        print("Fallo al leer el sensor")
    blynk.run()
    time.sleep(60)  # Enviar datos cada 60 segundos
```

Desarrollo de Dashboards

ThingSpeak:

- **Configuración de Gráficos:**
- Crear gráficos de línea para temperatura y humedad.
- Configurar actualizaciones automáticas y estilos visuales.
- **Widgets Avanzados:**
- Agregar indicadores de estado para mostrar si los dispositivos están encendidos o apagados.

Blynk:

- **Diseño de Tableros:**
- Añadir widgets de gráficos para visualizar datos de sensores.
- Configurar indicadores LED para mostrar el estado de luces y ventiladores.

Práctica:

- **Programación de Scripts en Python:**
- **Raspberry Pi:**
- Creación de scripts para leer datos de sensores y enviarlos a ThingSpeak.
- Integración con el script de reconocimiento de voz para una interacción más fluida.

- **Arduino con Netbook (Windows):**
- Desarrollo de scripts para enviar datos de sensores a Blynk.
- Utilización de módulos de comunicación (Wi-Fi o Ethernet) para la conexión a internet.
- **Visualización de Datos:**
- Configuración de gráficos y widgets en la plataforma IoT para monitorear temperatura y humedad en tiempo real.

Problemas y Soluciones:

- **Problema:** Latencia en la actualización de datos.
- **Solución:** Optimizar la frecuencia de actualización y asegurar una conexión estable a internet.
- **Problema:** Datos inconsistentes de sensores.
- **Solución:** Revisar conexiones y calibrar sensores nuevamente.

Semana 4: Ajustes Finales y Pruebas

Objetivo de la Etapa:

Integrar todos los sistemas (voz, IoT y control de dispositivos) y realizar pruebas completas para asegurar el correcto funcionamiento del modelo domótico.

Actividades:

Integración Total del Sistema

- **Unificación de Scripts de Reconocimiento de Voz y Monitoreo IoT:**
- Combinar los scripts de reconocimiento de voz y envío de datos a la plataforma IoT.
- Asegurar que los comandos de voz puedan activar dispositivos y que los sensores envíen datos sin interferencias.
- **Configuración de Comunicación entre Raspberry Pi/Netbook y Arduino:**
- **Raspberry Pi:**
- Utilizar puertos GPIO para comunicar directamente con dispositivos electrónicos.
- **Netbook con Arduino (Windows):**
- Utilizar comunicación serial para enviar comandos desde la netbook al Arduino.
- Asegurar que los scripts en Python manejan correctamente la comunicación bidireccional.

Pruebas de Funcionamiento

- **Verificación de la Operatividad de Todos los Sistemas:**
- Probar cada comando de voz y verificar que los dispositivos respondan adecuadamente.
- Verificar que los sensores envían datos correctos a la plataforma IoT.
- **Pruebas en Diferentes Condiciones Ambientales:**
- Realizar pruebas en entornos con diferentes niveles de ruido para evaluar la precisión del reconocimiento de voz.
- Verificar la precisión de los sensores en diferentes temperaturas y humedades.

Optimización y Corrección de Errores

- **Ajustes en los Scripts:**

- Mejorar el manejo de excepciones para evitar fallos en el script.
- Optimizar el código para reducir la latencia y mejorar la eficiencia.
- **Mejoras en el Hardware:**
- Reorganizar cables y componentes para evitar interferencias y asegurar una mejor conectividad.
- Implementar sistemas de disipación de calor si es necesario para mantener la estabilidad del hardware.

Práctica:

- **Simulación de Escenarios Reales:**
- Ejecutar comandos de voz en condiciones controladas y observar el comportamiento del sistema.
- Monitorear los datos en tiempo real y ajustar parámetros según sea necesario.
- **Feedback en Tiempo Real:**
- Utilizar la plataforma IoT para recibir retroalimentación sobre el estado del sistema.
- Ajustar los scripts y la configuración de sensores basándose en los datos recibidos.

Problemas y Soluciones:

- **Problema:** Comandos de voz inconsistentes.
- **Solución:** Mejorar el procesamiento de voz y ajustar el reconocimiento de comandos específicos.
- **Problema:** Fallos en la conexión de sensores.
- **Solución:** Revisar conexiones físicas y asegurar que los scripts están correctamente configurados para leer los datos.

5. Evaluación

5.1. Instrumento de Evaluación

- **Evaluación Continua:** Revisión semanal del progreso del proyecto y cumplimiento de objetivos.
- **Evaluación Final:** Verificación de la funcionalidad completa del modelo domótico, incluyendo reconocimiento de voz, control de dispositivos y monitoreo IoT.
- **Autoevaluación y Coevaluación:** Reflexión individual y evaluación del trabajo en equipo.

5.2. Evaluación Formativa

- **Observación Directa:** Monitoreo de la participación y compromiso de los estudiantes durante las actividades.
- **Bitácoras de Proyecto:** Registro de actividades, problemas y soluciones propuestas por los estudiantes.

5.3. Evaluación Sumativa

- **Modelo a Escala Funcional:** Evaluación de la integración y funcionamiento de todos los sistemas.
- **Presentación del Proyecto:** Exposición oral y demostración del modelo domótico.
- **Informe Final:** Documento escrito detallando el desarrollo del proyecto, decisiones técnicas y resultados obtenidos.

5.4. Entregables Esperables

- **Bitácoras de Proyecto:** Documentación diaria de actividades y avances.
- **Croquis del Modelo de Casa Domótica:** Plano 2D del modelo a escala.
- **Programas en Python:** Scripts desarrollados para el reconocimiento de voz y monitoreo IoT.
- **Documentación Técnica:** Especificaciones de hardware y software utilizados.
- **Informe Final:** Compilación de todo el trabajo realizado con análisis y conclusiones.

- **Presentación del Proyecto:** Diapositivas y demostración del modelo funcionando.

5.5. Rúbricas

Rúbrica para la Evaluación de la Integración del Sistema

Criterio	Nivel 1 (Insuficiente)	Nivel 2 (Aceptable)	Nivel 3 (Bueno)	Nivel 4 (Excelente)
Reconocimiento de Voz	No funciona o es inconsistente.	Funciona con dificultades.	Funciona correctamente con algunos ajustes.	Funciona de manera fluida y precisa.
Control de Dispositivos	No logra controlar los dispositivos.	Control básico con errores.	Control adecuado con mínima intervención.	Control completo y eficiente sin errores.
Monitoreo IoT	Datos no se envían o son incorrectos.	Datos se envían con errores o retrasos.	Datos se envían correctamente con pequeñas mejoras.	Datos se envían y visualizan en tiempo real sin fallas.
Integración del Sistema	No hay integración entre los sistemas.	Integración parcial con múltiples errores.	Integración adecuada con algunos aspectos por mejorar.	Integración completa y sin errores entre todos los sistemas.

Documentación	Documentación incompleta o ausente.	Documentación básica pero faltan detalles importantes.	Documentación clara y detallada con mínimos ajustes.	Documentación completa, clara y detallada.
Presentación	Presentación desorganizada y difícil de entender.	Presentación comprensible pero falta de claridad en algunos puntos.	Presentación clara y bien organizada con buena comunicación.	Presentación excepcionalmente clara, organizada y profesional.

6. Expansiones Opcionales

6.1. Agregar Más Dispositivos Controlados por Voz

- **Control de Persianas Automáticas:**
  Implementación de comandos para subir y bajar persianas.
- **Hardware:** Motores paso a paso o servomotores para controlar las persianas.
- **Software:** Añadir comandos como "Subir persianas" y "Bajar persianas" en los scripts de reconocimiento de voz.
- **Integración:** Conectar los motores al sistema de control mediante relés o drivers de motor.
- **Control de Puertas Automáticas:**
  Integración de sistemas de apertura y cierre de puertas mediante voz.
- **Hardware:** Actuadores lineales para abrir y cerrar puertas.
- **Software:** Programación de comandos como "Abrir puerta" y "Cerrar puerta".
- **Integración:** Configuración de relés para controlar los actuadores.

6.2. Integrar Notificaciones Push

- **Alertas en el Móvil:**
  Configuración de notificaciones para alertar al usuario sobre cambios en la temperatura, humedad o activación de dispositivos.
- **Herramientas:** Uso de servicios gratuitos como IFTTT para enviar notificaciones.
- **Implementación en Python:**

```
import requests


IFTTT_WEBHOOK_URL =
"https://maker.ifttt.com/trigger/{event}/with/key/{y
our_key}"

EVENT_NAME = "alerta_domotica"
```

```
def enviar_notificacion(mensaje):
    url = IFTTT_WEBHOOK_URL.format(event=EVENT_NAME,
your_key="TU_WEBHOOK_KEY")
    data = {"value1": mensaje}
    response = requests.post(url, json=data)
    if response.status_code == 200:
        print("Notificación enviada correctamente")
    else:
        print("Error al enviar la notificación")

# Ejemplo de uso
temperatura = 25
humedad = 60
if temperatura > 30:
    enviar_notificacion("Alerta: La temperatura es
demasiado alta.")
if humedad < 30:
    enviar_notificacion("Alerta: La humedad es
demasiado baja.")
```

- **Monitoreo de Anomalías:**
  Envío de alertas en caso de detección de condiciones anormales (temperatura demasiado alta o baja).

- **Implementación:** Añadir condiciones en los scripts de monitoreo para detectar anomalías y enviar notificaciones.
- **Ejemplo:**

```
if temperatura > 30 or temperatura < 15:
    enviar_notificacion(f"Alerta de temperatura: {temperatura}°C")
if humedad > 80 or humedad < 20:
    enviar_notificacion(f"Alerta de humedad: {humedad}%")
```

6.3. Implementar Reconocimiento Facial

- **Personalización del Sistema:**
  Uso de cámaras y algoritmos de reconocimiento facial para personalizar el control de dispositivos según el usuario.
- **Hardware:** Cámara USB compatible con Raspberry Pi o Arduino.
- **Software:** Uso de bibliotecas como OpenCV para el reconocimiento facial.
- **Implementación:**

```
import cv2


# Cargar el modelo pre-entrenado de reconocimiento facial
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

```
def reconocer_usuario(frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.1,
4)
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h),
(255, 0, 0), 2)
        return True  # Usuario reconocido
    return False

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if reconocer_usuario(frame):
        print("Usuario reconocido")
        # Implementar acciones específicas para el
usuario
    cv2.imshow('Reconocimiento Facial', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

```
cap.release()

cv2.destroyAllWindows()
```

- **Seguridad Adicional:**
  Implementación de sistemas de seguridad que permitan el acceso a ciertos dispositivos solo a usuarios reconocidos.
- **Implementación:** Vincular el reconocimiento facial con los comandos de voz para asegurar que solo usuarios autorizados puedan controlar dispositivos específicos.

7. Referencias

1. **Leopoldo Armesto.** *Programación de Robots Industriales | Sistemas Robotizados*. YouTube. 2020. https://www.youtube.com/watch?v=U48pJzoX0pU.
2. **OpenAI.** *Whisper: An Automatic Speech Recognition System*. 2023. https://openai.com/research/whisper.
3. **ThingSpeak.** *IoT Analytics Platform*. 2023. https://thingspeak.com/.
4. **Blynk.** *IoT Platform for Mobile Applications*. 2023. https://blynk.io/.
5. **Raspberry Pi Foundation.** *Getting Started with Raspberry Pi*. 2023. https://www.raspberrypi.org/documentation/.
6. **Adafruit.** *DHT Sensor Library*. 2023. https://learn.adafruit.com/dht.
7. **OpenCV.** *Open Source Computer Vision Library*. 2023. https://opencv.org/.
8. **Transformers by Hugging Face.** *Transformers Documentation*. 2023. https://huggingface.co/docs/transformers/index.
9. **IFTTT.** *IFTTT Webhooks Documentation*. 2023. https://ifttt.com/maker_webhooks.

Anexos

Anexo A: Diagramas de Conexión de Hardware

- **Diagrama 1:** Conexión de Relés a Raspberry Pi.
- **Diagrama 2:** Conexión de Relés a Arduino.
- **Diagrama 3:** Configuración de Sensores DHT11/DHT22.
- **Diagrama 4:** Integración de Cámara para Reconocimiento Facial.

Anexo B: Scripts de Python Utilizados

- **Script 1:** Reconocimiento de Voz y Control de Relés (Unificado para Raspberry Pi y Windows).
- **Script 2:** Envío de Datos a ThingSpeak.
- **Script 3:** Envío de Datos a Blynk.
- **Script 4:** Reconocimiento Facial con OpenCV.

Anexo C: Capturas de Pantalla de la Plataforma IoT

- **Captura 1:** Dashboard de ThingSpeak con Gráficos de Temperatura y Humedad.
- **Captura 2:** Dashboard de Blynk con Widgets de Monitoreo.
- **Captura 3:** Visualización en Tiempo Real de Datos en ThingSpeak.
- **Captura 4:** Indicadores de Estado en Blynk para Dispositivos Controlados.

Anexo D: Manual de Usuario del Modelo Domótico

- **Instrucciones de Inicio:**
- **Encendido del Sistema:** Cómo iniciar los scripts en Raspberry Pi o Netbook.
- **Uso de Comandos de Voz:** Lista de comandos disponibles y cómo activarlos.
- **Monitoreo de Datos:** Cómo acceder y interpretar los datos en la plataforma IoT.
- **Solución de Problemas:**
- **Problema:** Comandos de voz no reconocidos.
- **Solución:** Verificar la configuración del micrófono y la calibración del modelo de reconocimiento de voz.
- **Problema:** Sensores no envían datos a la plataforma IoT.

- **Solución:** Revisar conexiones físicas y asegurar que los scripts están corriendo correctamente.
- **Mantenimiento del Sistema:**
- **Actualización de Software:** Cómo actualizar los scripts y bibliotecas utilizadas.
- **Revisión de Hardware:** Inspección periódica de conexiones y componentes electrónicos.

Mejoras y Recomendaciones Adicionales (Futuras)

1. **Optimización del Reconocimiento de Voz:**
    - **Filtros de Audio:** Implementar filtros de ruido en el preprocesamiento del audio para mejorar la precisión del reconocimiento.
    - **Comandos Personalizados:** Crear un diccionario de comandos personalizados para adaptarse mejor al entorno específico del modelo a escala.

2. **Seguridad del Sistema:**
    - **Protección de Datos:** Implementar medidas para asegurar que los datos transmitidos a la plataforma IoT estén protegidos.
    - **Acceso Controlado:** Establecer mecanismos de autenticación para acceder a la plataforma IoT y al sistema de control de voz.

3. **Interfaz de Usuario Mejorada:**
    - **Aplicación Móvil:** Desarrollar una aplicación móvil personalizada para interactuar con el sistema domótico de manera más intuitiva.
    - **Alertas Visuales y Sonoras:** Integrar luces LED o sonidos que respondan a ciertos eventos en el sistema (por ejemplo, luces que parpadean cuando se recibe una alerta).

4. **Expansión de Funcionalidades:**
    - **Control de Otros Dispositivos:** Añadir control de electrodomésticos adicionales como termostatos inteligentes, sistemas de riego, etc.
    - **Automatización Basada en Tiempo:** Programar acciones automáticas según horarios predefinidos (por ejemplo, encender luces al atardecer).

5. **Documentación y Formación Continua:**
    - **Guías de Usuario Detalladas:** Crear manuales más extensos y detallados para facilitar el uso y mantenimiento del sistema.
    - **Capacitaciones Adicionales:** Ofrecer sesiones de formación continuas para los estudiantes sobre nuevas tecnologías y actualizaciones del sistema.

Links de actualización pequeños

Link pequeño

https://sor.bz/gQUf7

QR pequeño

Links de actualización completos

Link completo

https://github.com/guideahon/Pro_tec_2_IA_IOT_Domotica

QR completo