Evidencia de Aprendizaje N.º 1 Proyecto

Trabajo Práctico Fundamentos del IoT y Software

Enunciado del Proyecto: Desarrollo de Prototipos IoT con ESP32

Objetivo del Proyecto:
El objetivo de este proyecto es que los alumnos apliquen los conocimientos adquiridos en el curso mediante el desarrollo de dos prototipos utilizando la plataforma ESP32. Trabajando en grupos de no más de cuatro personas, los estudiantes deben seleccionar dos proyectos de la lista proporcionada, diseñar y construir los prototipos, y presentar un informe detallado y una demostración práctica en la plataforma Wokwi.

Formación de Grupos:

- Nahuel Argandoña
- Gastón Cane
- Eric Heredia

Informe Técnico: Detector de Movimiento con Alarma

1. Introducción

El proyecto del Detector de Movimiento con Alarma tiene como objetivo implementar un sistema de seguridad capaz de detectar movimiento en un área específica y activar una alarma cuando se detecte la presencia de personas. Este sistema utiliza un sensor PIR (infrarrojo pasivo) para la detección de movimiento y un ESP32 “Micro Python” para el procesamiento y la activación de la alarma.

---

2. Objetivos Específicos

- **Detectar Movimiento:** Utilizar el sensor PIR para detectar movimiento en el área.
- **Activar Alarma:** Encender un buzzer y un LED cuando se detecte movimiento.
- **Monitorear Estado:** Proveer información en la consola sobre el estado del área (presencia o ausencia de personas).
- ---

3. Descripción del Hardware Utilizado

- **Sensor PIR (Infrarrojo Pasivo):** Detecta cambios en la radiación infrarroja provocados por el movimiento de objetos en su campo de visión.

- **ESP32:** Microcontrolador que procesa las señales del sensor y controla el LED y el buzzer.
- **LED Rojo:** Señal visual que indica si hay movimiento.
- **Buzzer:** Genera una señal acústica para alertar sobre la detección de movimiento.
- **Resistor (1 kΩ):** Utilizado en el circuito del LED para limitar la corriente.
- ---

4. Esquemas de Conexión y Diagramas Eléctricos

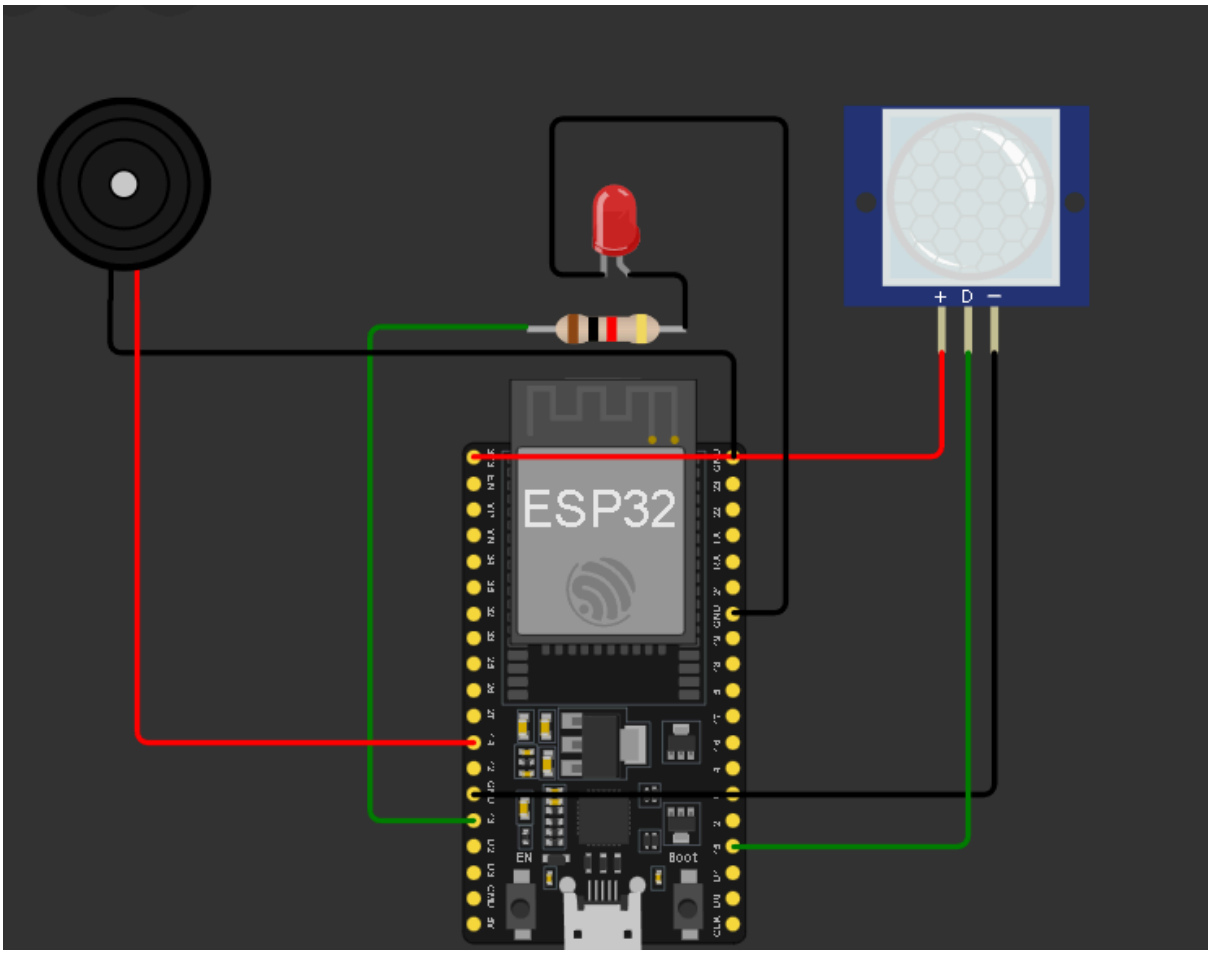
Link de workwi: https://wokwi.com/projects/406401817182924801

Diagrama de Conexión:

json

```
{
  "version": 1,
  "author": "Nahuel Argandoña",
  "editor": "wokwi",
  "parts": [
    {
      "type": "board-esp32-devkit-c-v4",
      "id": "esp",
      "top": -57.6,
      "left": -71.96,
      "attrs": { "env": "micropython-20231227-v1.22.0" }
    },
    { "type": "wokwi-pir-motion-sensor", "id": "pir1", "top":
-159.2, "left": 69.42, "attrs": {} },
    { "type": "wokwi-led", "id": "led1", "top": -138, "left": -34.6,
"attrs": { "color": "red" } },
    {
      "type": "wokwi-resistor",
      "id": "r1",
      "top": -82.45,
      "left": -48,
      "attrs": { "value": "1000" }
    },
    {
      "type": "wokwi-buzzer",
      "id": "bz1",
      "top": -170.4,
      "left": -228.6,
      "attrs": { "volume": "0.1" }
    }
```

```
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [] ],
    [ "esp:RX", "$serialMonitor:TX", "", [] ],
    [ "pir1:VCC", "esp:3V3", "red", [ "v0" ] ],
    [ "pir1:GND", "esp:GND.1", "black", [ "v0" ] ],
    [ "pir1:OUT", "esp:15", "green", [ "v0" ] ],
    [ "led1:A", "r1:2", "black", [ "v0", "h38.4" ] ],
    [ "led1:C", "esp:GND.3", "black", [ "v0", "h-18.8", "v-57.6",
"h86.4", "v96" ] ],
    [ "r1:1", "esp:13", "green", [ "v0", "h-57.6", "v163.2" ] ],
    [ "bz1:1", "esp:GND.2", "black", [ "v19.2", "h240" ] ],
    [ "bz1:2", "esp:14", "red", [ "v0" ] ]
  ],
  "dependencies": {}
}
```



---

5. Código Fuente

Código en MicroPython:

```
from machine import Pin  # Librería para manejo de pines
import utime  # Librería para el manejo del tiempo

# Crear el objeto del PIR, LED y buzzer
pir = Pin(15, Pin.IN, Pin.PULL_DOWN)  # PIR conectado al pin 15 y
GND
led_rojo = Pin(13, Pin.OUT)  # LED conectado al pin 13 (salida) y
GND
buzzer = Pin(14, Pin.OUT)  # Buzzer conectado al pin 14 (salida) y
GND

while True:
    estado = pir.value()  # Leer el valor del PIR (1 si detecta
movimiento, 0 si no)

    utime.sleep_ms(500)  # Espera de 500 ms entre lecturas

    if estado == 0:  # Si no detecta movimiento
        led_rojo.value(0)  # Apagar el LED
        buzzer.value(1)  # Encender el buzzer
        print('No hay nadie en el área')
    else:  # Si detecta movimiento
        led_rojo.value(1)  # Encender el LED
        buzzer.value(0)  # Apagar el buzzer
        print('Hay alguien en el área')

    utime.sleep(1)  # Espera de 1 segundo
```

---

6. Explicación del Funcionamiento del Código y Lógica de Programación

- **Inicialización:** El código inicializa los pines para el sensor PIR, el LED y el buzzer.
- **Lectura del Sensor PIR:** En un bucle continuo, se lee el valor del sensor PIR. Si el valor es `0`, significa que no se detecta movimiento. Si el valor es `1`, se detecta movimiento.
- **Control de Alarma:** Dependiendo del estado del sensor PIR, el código enciende o apaga el LED y el buzzer. Si no se detecta movimiento, el buzzer está encendido y el LED está apagado. Si se detecta movimiento, el buzzer está apagado y el LED está encendido.
- **Esperas:** Se incluyen esperas de 500 ms entre lecturas del sensor y 1 segundo entre ciclos del bucle para evitar lecturas demasiado frecuentes.

---

7. Resultados Obtenidos, Pruebas Realizadas y Posibles Mejoras Futuras

- **Resultados:** El sistema funciona según lo esperado. El LED y el buzzer responden correctamente a la detección de movimiento.
- **Pruebas Realizadas:** Se realizaron pruebas con el sensor PIR en diferentes condiciones de movimiento para verificar su precisión y respuesta.
- **Mejoras Futuras:**
    - **Integración de Notificación por Wi-Fi:** Implementar notificaciones por Wi-Fi para alertar sobre la detección de movimiento a través de una aplicación o servicio en la web.
    - **Ajustes del Sensor:** Calibrar el sensor PIR para mejorar la sensibilidad y reducir los falsos positivos.
    - **Interfaz de Usuario:** Desarrollar una interfaz de usuario para monitorear el estado del sistema en tiempo real.

8. Conclusiones

El proyecto del Detector de Movimiento con Alarma es efectivo para detectar presencia y activar una alarma en respuesta a la detección de movimiento. La implementación actual cumple con los objetivos básicos del proyecto, pero se pueden agregar características adicionales, como notificaciones por Wi-Fi, para mejorar su funcionalidad. Las pruebas realizadas confirmaron que el sistema es confiable y responde adecuadamente a las condiciones de movimiento.

Informe Técnico: control de iluminación automático

---

1. Introducción a los Proyectos Seleccionados

Este informe técnico presenta un proyecto de control de iluminación basado en un sensor LDR (resistor dependiente de la luz) y un LED controlado mediante PWM utilizando un

microcontrolador ESP32. El proyecto tiene como objetivo demostrar cómo medir la intensidad de luz ambiente y ajustar el brillo de un LED en función de esa medición.

---

2. Objetivos Específicos del Proyecto

1. **Medición de la Intensidad de Luz**: Utilizar un sensor LDR para medir la cantidad de luz ambiental.
2. **Control del Brillo del LED**: Ajustar el brillo de un LED en función de la intensidad de luz medida.
3. **Desarrollo de un Sistema de Control Basado en Umbrales**: Implementar umbrales para determinar el nivel de luz y modificar el brillo del LED.
4. **Implementación de Lecturas y Control en Tiempo Real**: Realizar lecturas periódicas del sensor y ajustar el LED en tiempo real.

---

3. Descripción del Hardware Utilizado

- **ESP32 Devkit C v4**: Microcontrolador con capacidades Wi-Fi y Bluetooth, utilizado para manejar el sensor LDR y el LED.
- **LED Rojo (wokwi-led)**: Actuador que emite luz y cuyo brillo se controla mediante PWM.
- **Resistor (wokwi-resistor)**: Resistor de 1000 ohmios, utilizado para limitar la corriente al LED.
- **Sensor LDR (wokwi-photoresistor-sensor)**: Sensor que mide la intensidad de luz ambiental.

---

4. Esquemas de Conexión y Diagramas Eléctricos

Link de wokwi: https://wokwi.com/projects/408179552540136449

Diagrama de Conexiones:

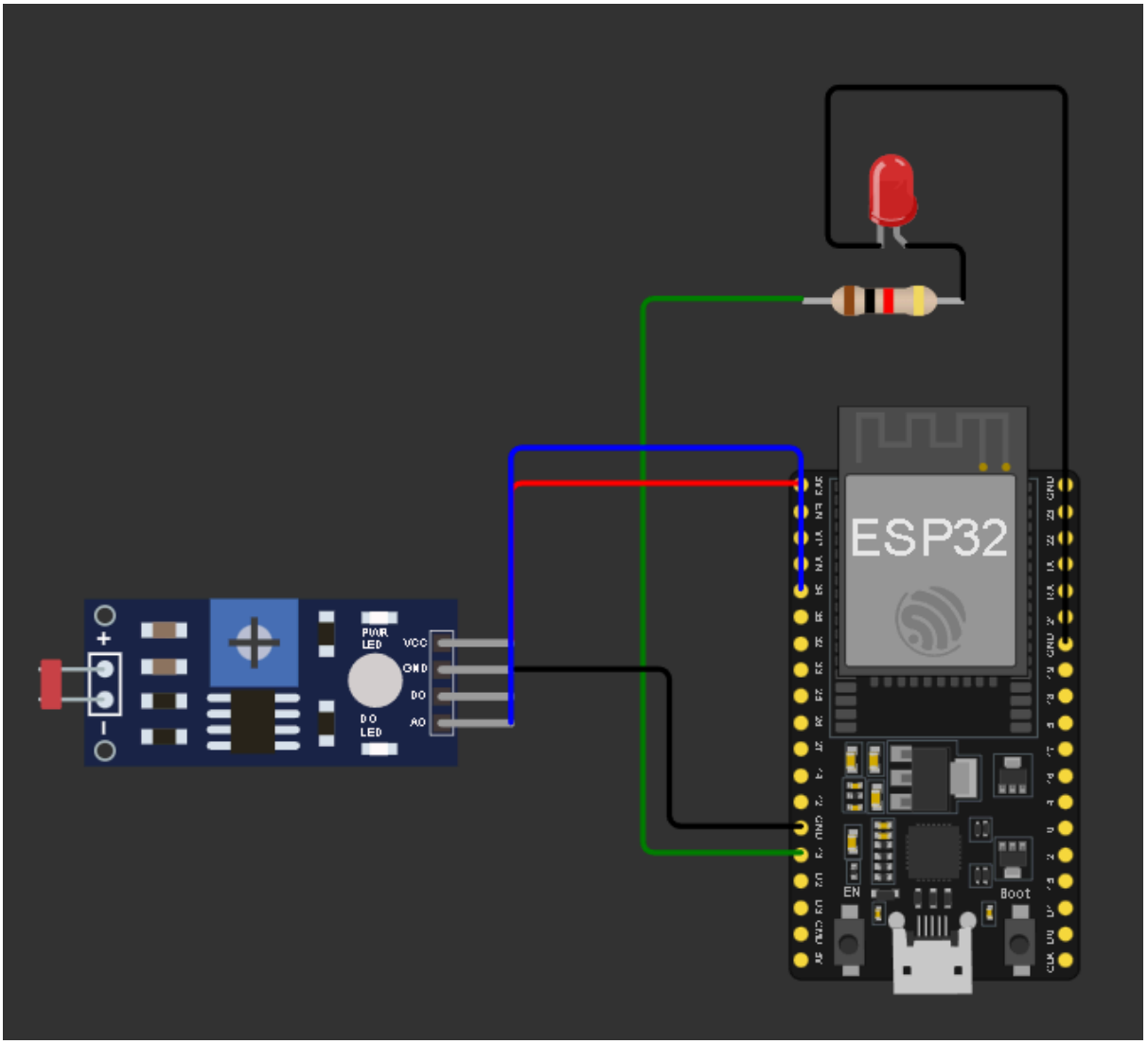
- **LED Rojo**:
  - Anodo (A) conectado al pin 13 del ESP32 a través del resistor (R1).
  - Cátodo (C) conectado a GND del ESP32.
- **Resistor (R1)**:
  - Conectado entre el pin 13 del ESP32 y el ánodo del LED.
- **Sensor LDR**:
  - VCC conectado al pin 3V3 del ESP32.
  - GND conectado al GND del ESP32.
  - Salida analógica (AO) conectada al pin 34 del ESP32.

Diagrama del JSON:

```
{
```

```
  "version": 1,
  "author": "Nahuel Argandoña",
  "editor": "wokwi",
  "parts": [
    {
      "type": "board-esp32-devkit-c-v4",
      "id": "esp",
      "top": -38.4,
      "left": -52.76,
      "attrs": { "env": "micropython-20231227-v1.22.0" }
    },
    { "type": "wokwi-led", "id": "led1", "top": -138, "left": -34.6,
"attrs": { "color": "red" } },
    {
      "type": "wokwi-resistor",
      "id": "r1",
      "top": -82.45,
      "left": -48,
      "attrs": { "value": "1000" }
    },
    { "type": "wokwi-photoresistor-sensor", "id": "ldr1", "top": 32,
"left": -325.6, "attrs": {} }
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [] ],
```

```
    [ "esp:RX", "$serialMonitor:TX", "", [] ],

    [ "led1:A", "r1:2", "black", [ "v0", "h38.4" ] ],

    [ "led1:C", "esp:GND.3", "black", [ "v0", "h-18.8", "v-57.6",
"h86.4", "v96" ] ],

    [ "r1:1", "esp:13", "green", [ "v0", "h-57.6", "v163.2" ] ],

    [ "ldr1:VCC", "esp:3V3", "red", [ "h0", "v-163.2" ] ],

    [ "ldr1:GND", "esp:GND.1", "black", [ "h57.6", "v-38.8" ] ],

    [ "ldr1:AO", "esp:34", "blue", [ "h0", "v-100", "h200" ] ]

  ],

  "dependencies": {}

}
```



---

5. Código Fuente Detallado en MicroPython

```
from machine import Pin, ADC, PWM  # Librería para manejo de pines, ADC y PWM
import utime  # Librería para manejo del tiempo

# Crear el objeto del LED con PWM y sensor LDR
led_rojo = PWM(Pin(13), freq=1000)  # LED en el pin 13 con una frecuencia de 1000 Hz
ldr = ADC(Pin(34))  # LDR conectado al pin 34 (entrada analógica)
ldr.atten(ADC.ATTN_11DB)  # Configurar el rango de lectura del LDR (0-3.3V)

# Umbrales para determinar los niveles de luz
umbral_bajo = 1000  # Ajusta este valor según el entorno
```

```
umbral_medio = 3000

while True:
    valor_ldr = ldr.read()  # Leer el valor del LDR (rango de 0 a
4095)

    print(f"Valor LDR: {valor_ldr}")  # Mostrar el valor del LDR en
consola para debug

    if valor_ldr > umbral_medio:  # Si hay mucha luz, apagar el LED
        led_rojo.duty(1023)  # LED apagado
        print("Luz alta - LED apagado")

    elif umbral_bajo < valor_ldr <= umbral_medio:  # Si hay luz
media, ajustar brillo bajo del LED
        led_rojo.duty(512)  # LED con brillo bajo (rango PWM de 0 a
1023)
        print("Luz media - LED con brillo bajo")

    else:  # Si hay poca luz o está oscuro, ajustar brillo alto del
LED
        led_rojo.duty(0)  # LED con brillo alto (rango PWM de 0 a
1023)
        print("Luz baja - LED con brillo alto")

    utime.sleep(1)  # Espera de 1 segundo entre lecturas
```

---

6. Explicación del Funcionamiento del Código y la Lógica de Programación

El código realiza las siguientes funciones:

- **Inicialización**: Configura el LED para usar PWM y el sensor LDR para lectura analógica.
- **Lectura del Sensor**: Mide la luz ambiental con el sensor LDR.
- **Control del LED**: Ajusta el brillo del LED basándose en los valores leídos del LDR:
    - Si la luz es alta (valor LDR > umbral_medio), el LED está apagado.
    - Si la luz es media (umbral_bajo < valor LDR <= umbral_medio), el LED tiene un brillo bajo.
    - Si la luz es baja (valor LDR <= umbral_bajo), el LED tiene un brillo alto.
- **Espera**: Pausa de 1 segundo entre lecturas para evitar lecturas excesivas.

---

7. Resultados Obtenidos, Pruebas Realizadas y Posibles Mejoras Futuras

- **Resultados Obtenidos**: El sistema ajusta el brillo del LED correctamente según la cantidad de luz ambiental. El LED cambia de estado (encendido/apagado) y el brillo se ajusta en función de los umbrales predefinidos.
- **Pruebas Realizadas**:
    - Se verificó el funcionamiento del LED en diferentes niveles de luz.
    - Se comprobó la correcta lectura del sensor LDR y su impacto en el brillo del LED.
- **Posibles Mejoras**:
    - **Calibración de Umbrales**: Ajustar los umbrales para adaptarse a diferentes condiciones ambientales.
    - **Interfaz de Usuario**: Añadir una interfaz para ajustar los umbrales y otros parámetros en tiempo real.
    - **Optimización del Código**: Implementar técnicas de debouncing o filtrado para mejorar la estabilidad de las lecturas del LDR.

---

8. Conclusiones

El proyecto demostró con éxito el uso de un sensor LDR y un LED controlado por PWM para crear un sistema de control de iluminación basado en la intensidad de luz ambiental. La implementación en MicroPython permitió un desarrollo rápido y flexible del sistema, y el hardware utilizado se demostró adecuado para el propósito. Las mejoras futuras pueden centrarse en la calibración y la expansión de funcionalidades para un control más avanzado y una mayor adaptabilidad.

Link wokwi de proyectos echo de los videos de clase:

https://wokwi.com/projects/406399818014988289 "led on and off python"

https://wokwi.com/projects/406400729259051009 "semaforo python"

https://wokwi.com/projects/406407331074018305 "sensor de movimiento 2 python esp32"

https://wokwi.com/projects/406646361165009921"c+ led prendido"

https://wokwi.com/projects/406665075593823233 "c+ led automático efecto desvaneciendo"

https://wokwi.com/projects/406661280389997569 "c+ led con tensiometro"

https://wokwi.com/projects/406664894926261249 "c+ pulsador led"