

TECNICATURA SUPERIOR EN Desarrollo del Software

Aproximación al Mundo del Trabajo

Profesor Mainero Alejandro

05 de octubre de 2024

- **Título del trabajo: Monitoreo de Movimiento IoT**
- **Nombre de los estudiantes**
- **[Nahuel Argandoña] [Gastón Cané] [Eric Heredia]**

Índice

1. Introducción	3
2. Objetivos	4
2.1. Objetivo General	4
2.2. Objetivos Específicos	4
3. Metodología	5
3.1. Diseño del Sistema	5
3.2. Conexión a la Base de Datos	6
4. Desarrollo	7
4.1. Código del Microcontrolador	7
4.2. Código de la API con Flask	9
5. Resultados	11
6. Conclusiones	12
7. Anexos	13
7.1. Diagramas del Sistema	13
7.2. Documentación Adicional	14

1. Introducción

El presente trabajo aborda el desarrollo de un sistema de monitoreo de movimiento utilizando un sensor PIR, un microcontrolador y una API para almacenar y recuperar datos en tiempo real. Se detalla la conexión a la base de datos y el proceso de envío de datos mediante solicitudes HTTP.

2. Objetivos

2.1. Objetivo General

Desarrollar un sistema de monitoreo que permita detectar movimiento y almacenar la información en una base de datos en la nube para su posterior análisis.

2.2. Objetivos Específicos

- Implementar un microcontrolador con un sensor PIR para detectar movimiento.
- Configurar una API RESTful utilizando Flask para gestionar la interacción con la base de datos.
- Almacenar y recuperar datos desde una base de datos MySQL en Clever Cloud.

3. Metodología

3.1. Diseño del Sistema

El sistema está compuesto por un microcontrolador que se conecta a una red Wi-Fi y utiliza un sensor PIR para detectar movimiento. Cuando se detecta movimiento, se envían datos a una API, que a su vez interactúa con una base de datos para almacenar la información.

3.2. Conexión a la Base de Datos

Se utiliza la biblioteca `mysql.connector` para gestionar la conexión con la base de datos y realizar operaciones de inserción y recuperación de datos.

4. Desarrollo

4.1. Código del Microcontrolador

```
from machine import Pin # Librería para manejo de pines
import utime # Librería para el manejo del tiempo
import network # Librería para conectividad de red
import urequests # Librería para hacer solicitudes HTTP

# Datos de Wi-Fi para Wokwi
SSID = 'Wokwi-GUEST' # Red Wi-Fi simulada en Wokwi
PASSWORD = '' # Sin contraseña en la red simulada

# Conectar a Wi-Fi
def conectar_wifi(ssid, password):
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True) # Activar el modo estación
```

```

wlan.connect(ssid, password) # Conectar a la red Wi-Fi

# Esperar a que se conecte
while not wlan.isconnected():
    print('Conectando a la red...')
    utime.sleep(1)

    print('Conectado a:', wlan.ifconfig()) # Mostrar dirección IP y
detalles de conexión

# 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

# Conectar a la red Wi-Fi simulada
conectar_wifi(SSID, PASSWORD)

# Función para enviar datos al servidor
def enviar_datos(sensor, valor, message):

    data = {'sensor': sensor, 'value': valor, 'message': message}
    print('Enviando datos:', data)

    try:
        response =
requests.post('https://iot-lot-2.onrender.com/data', json=data) #
Cambia la URL si es necesario
        print('Código de respuesta:', response.status_code)
        print('Respuesta JSON:', response.json())
    except Exception as e:
        print('Error en la solicitud:', e)

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

    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')

```

```

        enviar_datos('PIR', 0, 'No se detecta movimiento cercano') #
Enviar valor de 0 al servidor
    else: # Si detecta movimiento
        led_rojo.value(1) # Encender el LED
        buzzer.value(0) # Apagar el buzzer
        print('Hay alguien en el área')
        enviar_datos('PIR', 1, 'Se detecta movimiento cercano') #
Enviar valor de 1 al servidor

    utime.sleep(2) # Espera de 2 segundos

```

4.2. Código de la API con Flask

python

```

from flask import Flask, request, jsonify
import mysql.connector

app = Flask(__name__)

# Configuración de la conexión a la base de datos
db_config = {
    "host": "bxy5ofa8ezud0x0caavs-mysql.services.clever-cloud.com",
    "user": "uupemiqzelzhijs",
    "password": "XnExGlv7QzWuydrfjtLK", # Reemplaza con tu contraseña
    "database": "bxy5ofa8ezud0x0caavs",
}

# Ruta para insertar datos
@app.route("/data", methods=["POST"])
def insert_data():
    if request.is_json:
        data = request.get_json()
        sensor = data.get("sensor")
        value = data.get("value")
        message = data.get("message")

        if sensor is None or value is None:
            return jsonify({"status": "error", "message": "Both 'sensor' and 'value' are required and 'message' are required"}), 400

        try:

```

```

        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor()

        cursor.execute("INSERT INTO sensor_data (sensor, value,
message) VALUES (%s, %s, %s)", (sensor, value, message))
        conn.commit()

        cursor.close()
        conn.close()

        return jsonify({"status": "success"}), 201

    except mysql.connector.Error as err:
        return jsonify({"status": "error", "message": str(err)}),
500
    else:
        return jsonify({"status": "error", "message": "Request must be
JSON"}), 400

# Ruta para mostrar los datos almacenados
@app.route("/data", methods=["GET"])
def get_data():
    try:
        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor()

        # Ejecutar consulta para obtener todos los datos
        cursor.execute("SELECT id, sensor, value, message, timestamp
FROM sensor_data")
        rows = cursor.fetchall()

        # Cerrar el cursor y la conexión
        cursor.close()
        conn.close()

        # Formatear los datos como una lista de diccionarios
        results = []
        for row in rows:
            results.append({
                "id": row[0],
                "sensor": row[1],
                "value": row[2],

```

```

        "message": row[3],
        "timestamp": row[4].strftime("%Y-%m-%d %H:%M:%S") #
Formato de fecha
    })

    return jsonify({"status": "success", "data": results}), 200

except mysql.connector.Error as err:
    return jsonify({"status": "error", "message": str(err)}), 500

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)

```

Requirements

```

Flask #Flask: El marco web que estás usando para desarrollar la API.
mysql-connector-python

# pip install -r requirements.txt para instalar las dependencias.

# Iniciar el Servidor Flask
# export FLASK_APP=app.py      # Para Linux o macOS
#set FLASK_APP=app.py          # Para Windows
# python app.py # vs code windows

#export FLASK_ENV=development  # (opcional) para habilitar el modo de
desarrollo
#flask run                      # Inicia el servidor

```

MYSQL DATA BASE

MySQL Workbench

iot_lot x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

Administration Schemas Information

No object selected

Query 1 x sensor_data

Limit to 1000 rows

```
1 • CREATE DATABASE iot_data;
2
3 • USE iot_data;
4
5 • CREATE TABLE sensor_data (
6     id INT AUTO_INCREMENT PRIMARY KEY,
7     sensor VARCHAR(50),
8     value FLOAT,
9     timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
10 );
11
12 • ALTER TABLE sensor_data
13     ADD COLUMN message VARCHAR(50) AFTER value;
14
15 • TRUNCATE TABLE sensor_data;
```

Output

Action Output

#	Time	Action
---	------	--------

CONSULTA

The screenshot displays the MySQL Workbench interface. The 'Query 1' tab is active, showing the SQL query: `SELECT * FROM bxy5ofa8ezud0x0caavs.sensor_data;`. The 'Result Grid' is visible at the bottom, showing the results of the query. The results are as follows:

	id	sensor	value	message	timestamp
▶	1	PIR	0	No se detecta movimiento cercano	2024-10-03 17:21:57
	2	PIR	0	No se detecta movimiento cercano	2024-10-03 17:22:07
	3	PIR	1	Se detecta movimiento cercano	2024-10-03 17:22:17
	4	PIR	0	No se detecta movimiento cercano	2024-10-03 17:25:02
	5	PIR	0	No se detecta movimiento cercano	2024-10-03 17:25:14
	6	PIR	1	Se detecta movimiento cercano	2024-10-03 17:26:28
*	NULL	NULL	NULL	NULL	NULL

5. Resultados

Los resultados del proyecto muestran la efectividad del sistema en la detección de movimiento y el correcto almacenamiento de datos en la base de datos.

6. Conclusiones

El sistema desarrollado cumple con los objetivos planteados, demostrando la viabilidad de soluciones IoT para el monitoreo en tiempo real.

7. Anexos

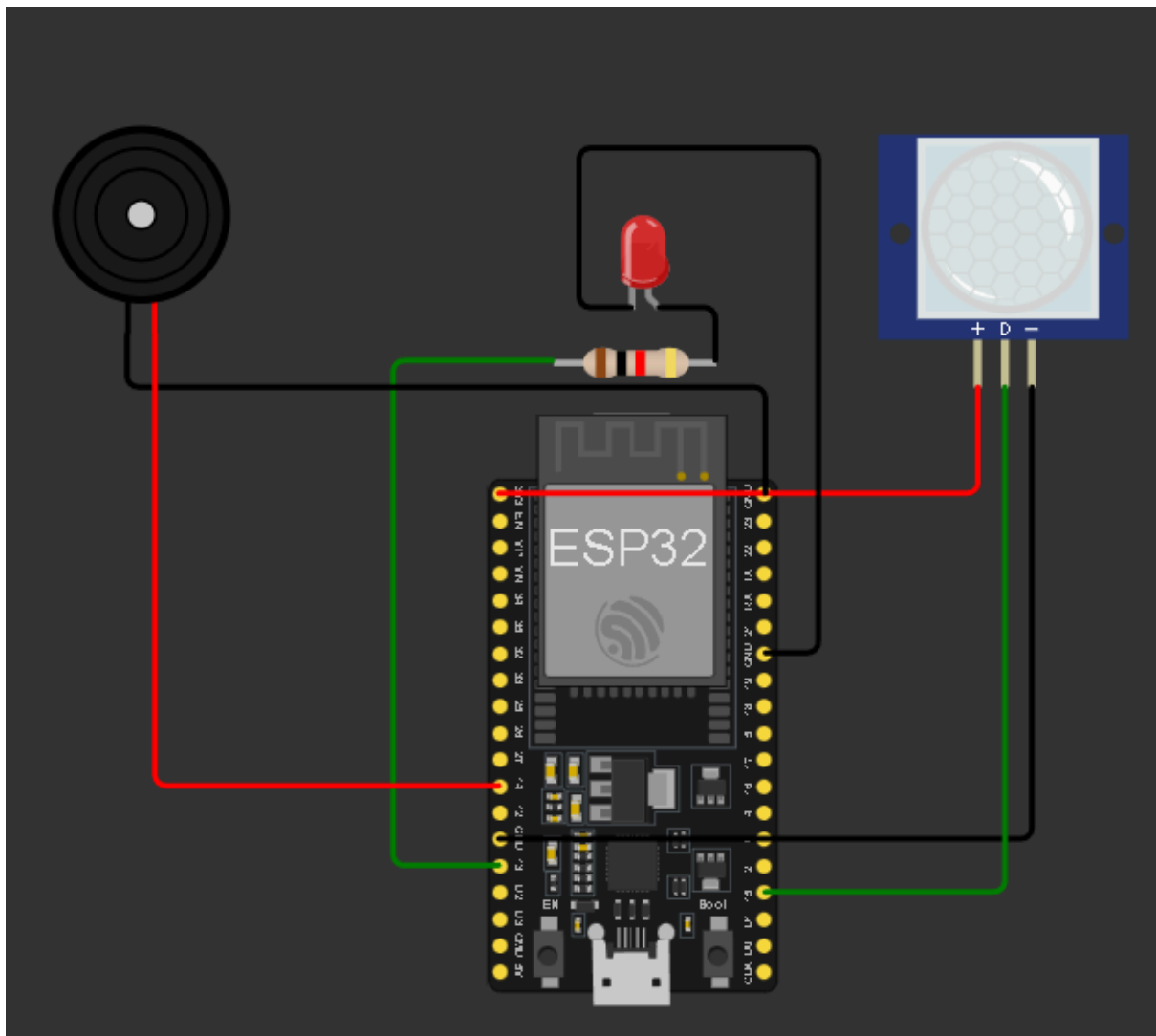
7.1. Diagramas del Sistema

```
{
  "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": {}
}

```



7.2. Documentación Adicional

<https://www.clever-cloud.com/product/mysql/>

<https://docs.render.com/deploy-flask>

<https://wokwi.com/projects/406401817182924801>

https://github.com/Aubar48/iot_lot_2