import paho.mqtt.client as mqtt

import mysql.connector

import json

# MQTT configuration

MQTT\_BROKER\_HOST = "broker.emqx.io"

MQTT\_BROKER\_PORT = 1883

MQTT\_TOPIC = "Sensor"

# MySQL configuration

MYSQL\_HOST = "localhost"

MYSQL\_USER = "root"

MYSQL\_PASSWORD = ""

MYSQL\_DATABASE = "ALL\_\_sensor"

# MQTT callback when a message is received

def on\_message(client, userdata, message):

payload = message.payload.decode("utf-8")

insert\_data(payload)

# Function to create the database if it doesn't exist

def create\_database\_if\_not\_exists():

try:

# Connect to MySQL server without specifying the database

connection = mysql.connector.connect(

host=MYSQL\_HOST,

user=MYSQL\_USER,

password=MYSQL\_PASSWORD

)

# Create a cursor object to execute SQL queries

cursor = connection.cursor()

# Define the SQL query to create the database if it doesn't exist

create\_database\_query = f"CREATE DATABASE IF NOT EXISTS {MYSQL\_DATABASE}"

# Execute the SQL query to create the database

cursor.execute(create\_database\_query)

# Close the cursor and the database connection

cursor.close()

connection.close()

print(f"Database '{MYSQL\_DATABASE}' created or already exists")

except Exception as e:

print(f"Error: {str(e)}")

# Create the database if it doesn't exist

create\_database\_if\_not\_exists()

def create\_table\_if\_not\_exists():

try:

# Connect to the MySQL database

connection = mysql.connector.connect(

host=MYSQL\_HOST,

user=MYSQL\_USER,

password=MYSQL\_PASSWORD,

database=MYSQL\_DATABASE

)

# Create a cursor object to execute SQL queries

cursor = connection.cursor()

# Define the SQL query to create the table if it doesn't exist

create\_table\_query = """

CREATE TABLE IF NOT EXISTS sensor\_data (

id INT AUTO\_INCREMENT PRIMARY KEY,

temperature FLOAT,

humidity FLOAT,

ultra\_sonic1sensor INT,

ultra\_sonic2sensor INT,

LDRvalue INT,

Gasvalue FLOAT,

timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

)

"""

# Print the query to check if it's correct

print("Creating table with query:")

print(create\_table\_query)

# Execute the SQL query to create the table

cursor.execute(create\_table\_query)

# Commit the transaction

connection.commit()

print("Table created successfully")

# Close the cursor and the database connection

cursor.close()

connection.close()

except Exception as e:

print(f"Error: {str(e)}")

# Create the table if it doesn't exist

create\_table\_if\_not\_exists()

# Function to insert data into MySQL

def insert\_data(data):

try:

# Split the received data by comma to extract individual values

values = data.split(',')

# Check if there are enough values

if len(values) == 6:

# Extract values

temperature = float(values[0])

humidity = float(values[1])

ultra\_sonic1sensor = int(values[2])

ultra\_sonic2sensor = int(values[3])

LDRvalue = int(values[5])

Gasvalue = float(values[4])

# Create a dictionary to store the data

data\_dict = {

"humidity": humidity,

"temperature": temperature,

"ultra\_sonic1sensor": ultra\_sonic1sensor,

"ultra\_sonic2sensor": ultra\_sonic2sensor,

"Gasvalue": Gasvalue,

"LDRvalue": LDRvalue,

}

# Convert the dictionary to a JSON string

json\_data = json.dumps(data\_dict)

# Connect to the MySQL database

connection = mysql.connector.connect(

host=MYSQL\_HOST,

user=MYSQL\_USER,

password=MYSQL\_PASSWORD,

database=MYSQL\_DATABASE

)

# Create a cursor object to execute SQL queries

cursor = connection.cursor()

# Define the SQL query to insert data into the database

sql\_insert\_query = "INSERT INTO sensor\_data (humidity,temperature,ultra\_sonic1sensor,ultra\_sonic2sensor,Gasvalue,LDRvalue) VALUES (%s, %s, %s, %s, %s, %s)"

# Execute the SQL query with data parameters

cursor.execute(sql\_insert\_query, (humidity,temperature,ultra\_sonic1sensor,ultra\_sonic2sensor,Gasvalue,LDRvalue))

# Commit the transaction

connection.commit()

print(f"Data inserted: humidity={humidity},temperature={temperature}, ultra\_sonic1sensor={ultra\_sonic1sensor},ultra\_sonic2sensor={ultra\_sonic2sensor}, Gasvalue={Gasvalue},LDRvalue={LDRvalue}, ")

# Close the cursor and the database connection

cursor.close()

connection.close()

else:

print("Invalid data format: Expected 5 values separated by commas")

except Exception as e:

print(f"Error: {str(e)}")

# MQTT client setup

client = mqtt.Client()

client.on\_message = on\_message

# Connect to MQTT broker

client.connect(MQTT\_BROKER\_HOST, MQTT\_BROKER\_PORT, 60)

client.subscribe(MQTT\_TOPIC)

# Start the MQTT loop

client.loop\_forever()