```python
#!/usr/bin/env python3
"""
محاكي جهاز مراقبة حسّاسات ذكي - Simulation Program
مزايا:
- محاكاة اهتزاز وحرارة وصوت مع سيناريوهات
- تحليل المخاطر وتصنيفها
- تسجيل نتائج إلى CSV وConsole
- إحصاءات ملخّصة بنهاية التشغيل

تشغيل مثال:
python3 simulator.py --duration 60 --interval 1 --scenario random --csv output.csv
"""

import argparse
import asyncio
import csv
import logging
import random
import statistics
from datetime import datetime, timezone

# -------------------------
# إعدادات افتراضية
# -------------------------
DEFAULT_INTERVAL = 2.0  # ثانية بين قراءات الحساسات
DEFAULT_DURATION = 30   # مدة المحاكاة بالثواني
LOG_FORMAT = "%(asctime)s | %(levelname)s | %(message)s"

# -------------------------
# وظائف قراءة الحساسات (قابلة للتعديل/الاستبدال)
# -------------------------
def read_vibration(scenario="normal"):
# اهتزاز: normal <5, attack >5
    if scenario == "attack":
```

```
        return random.uniform(5.5, 12.0)
    if scenario == "fire":
# اهتزاز عادي
        return random.uniform(1.0, 6.0)
    # normal or random
    return random.uniform(1.0, 7.5)

def read_temperature(scenario="normal"):
# حرارة: normal <35, fire >40
    if scenario == "fire":
        return random.uniform(40.5, 70.0)
    if scenario == "attack":
        return random.uniform(30.0, 38.0)
    return random.uniform(30.0, 45.0)

def read_acoustic(scenario="normal"):
# صوت: normal <50, attack >60
    if scenario == "attack":
        return random.uniform(60.0, 120.0)
    if scenario == "fire":
        return random.uniform(30.0, 70.0)
    return random.uniform(20.0, 80.0)

# -------------------------
# تحليل المخاطر
# -------------------------
def analyze_risk(vib, temp, sound):
    """
ترجع (risk_type, probability, level)
    risk_type: "لا يوجد خطر" أو "Attempted Break-in" أو "Fire Risk" أو "Unknown"
    probability: 0-100
    level: Low/Medium/High
    """
    risk_type = "لا يوجد خطر"
    probability = 0
    level = "Low"

# محاولة كسر: اهتزاز عالي + صوت عالي
```

```
    if vib > 5.0 and sound > 60:
        risk_type = "Attempted Break-in"
# مزيج من الاهتزاز والصوت يعطينا احتمالًا
        probability = int(min((vib * 10) + (sound * 0.5), 100))
        level = "High" if probability > 65 else "Medium"

# حريق: درجة حرارة مرتفعة
    elif temp > 40.0:
        risk_type = "Fire Risk"
        probability = int(min((temp - 35) * 10, 100))
        level = "High" if probability > 70 else "Medium"

    else:
# حالة عادية: احتمال يعتمد على الاهتزاز (كمثال)
        probability = int(min(vib * 8, 100))
        level = "Low"

# ضمان القيم داخل النطاق
    probability = max(0, min(100, probability))

    return risk_type, probability, level

# -------------------------
# محاكاة إرسال تنبيه (mock)
# -------------------------
async def send_alert_mock(risk_type, probability, level, transport="LoRaWAN"):
# ممكن هنا تربط مع مكتبة إرسال فعلية (MQTT, HTTP, LoRa, SMS...)
    await asyncio.sleep(0.05)  # محاكاة تأخير الشبكة
    return f"Alert sent via {transport}: {risk_type} | {probability}% | {level}"

# -------------------------
# حلقة المحاكاة الأساسية (async)
# -------------------------
async def run_simulation(duration, interval, scenario, csvfile, verbose, transport):
    logger = logging.getLogger("simulator")
    start_time = datetime.now(timezone.utc)
    end_time = start_time.timestamp() + duration if duration > 0 else None
```

```
# إحصاءات
    readings = []
    alerts = []
    counters = {"Attempted Break-in": 0, "Fire Risk": 0, "لا يوجد خطر": 0, "Unknown": 0}

# إعداد CSV إذا طُلب
    csv_writer = None
    csv_fh = None
    if csvfile:
        csv_fh = open(csvfile, "w", newline="", encoding="utf-8")
        csv_writer = csv.writer(csv_fh)
        csv_writer.writerow(["timestamp_utc", "vibration", "temperature", "acoustic", "risk_type", "probability", "level"])

    iteration = 0
    try:
        while True:
            now = datetime.now(timezone.utc)
            if end_time and now.timestamp() >= end_time:
                break

            iteration += 1
# قراءات الحساسات
            vib = read_vibration(scenario)
            temp = read_temperature(scenario)
            sound = read_acoustic(scenario)

            risk_type, probability, level = analyze_risk(vib, temp, sound)

# تسجيل إحصائيات محلية
            readings.append({"vib": vib, "temp": temp, "sound": sound, "prob": probability})
            counters[risk_type] = counters.get(risk_type, 0) + 1

# طباعة أو لوج
            msg = f"[{now.isoformat()}] Read #{iteration} | V={vib:.2f} | T={temp:.2f} | S={sound:.2f} => {risk_type} ({probability}%) [{level}]"
            if probability > 50 and risk_type != "لا يوجد خطر":
# إرسال تنبيه محاكاة
```

```
                alert_result = await send_alert_mock(risk_type, probability, level, transport=transport)
                alerts.append({"time": now.isoformat(), "type": risk_type, "prob": probability, "level": level})
                logger.warning(msg + " ---> ALERT! " + alert_result)
            else:
                if verbose:
                    logger.info(msg)
                else:
                    logger.debug(msg)

# كتابة CSV
            if csv_writer:
                csv_writer.writerow([now.isoformat(), f"{vib:.2f}", f"{temp:.2f}", f"{sound:.2f}", risk_type, probability, level])

            await asyncio.sleep(interval)

    except asyncio.CancelledError:
        logger.info("Simulation cancelled.")
    finally:
        if csv_fh:
            csv_fh.close()

# ملخص إحصائي
    summary = {
        "total_readings": len(readings),
        "alerts_sent": len(alerts),
        "counters": counters,
        "max_vibration": max(r["vib"] for r in readings) if readings else None,
        "max_temperature": max(r["temp"] for r in readings) if readings else None,
        "max_acoustic": max(r["sound"] for r in readings) if readings else None,
        "avg_vibration": statistics.mean(r["vib"] for r in readings) if readings else None,
        "avg_temperature": statistics.mean(r["temp"] for r in readings) if readings else None,
        "avg_acoustic": statistics.mean(r["sound"] for r in readings) if readings else None,
    }
    return summary

# -------------------------
```

```python
# واجهة السطر (CLI)
# -------------------------
def parse_args():
    p = argparse.ArgumentParser(description="محاكي جهاز المراقبة الذكي - Simulation")
    p.add_argument("--duration", "-d", type=int, default=DEFAULT_DURATION, help="مدة المحاكاة بالثواني (0 = إلى أن توقف يدويًا)")
    p.add_argument("--interval", "-i", type=float, default=DEFAULT_INTERVAL, help="الفاصل الزمني بين قراءات الحساسات بالثواني")
    p.add_argument("--scenario", "-s", type=str, default="random", choices=["random", "normal", "attack", "fire"], help="نمط توليد البيانات")
    p.add_argument("--csv", type=str, default=None, help="مسار ملف CSV لتسجيل القراءات")
    p.add_argument("--verbose", action="store_true", help="طباعة كافة القراءات (تفصيلي)")
    p.add_argument("--log", type=str, default=None, help="مسار ملف لوج (إذا لم يُعطَ، يطبع في الطرفية)")
    p.add_argument("--seed", type=int, default=None, help="بذرة عشوائية لإعادة إنتاج النتائج")
    p.add_argument("--transport", type=str, default="LoRaWAN", help="نوع النقل لمحاكاة الإرسال (LoRaWAN/5G/MQTT)")
    return p.parse_args()

# -------------------------
# نقطة البداية
# -------------------------
def main():
    args = parse_args()

    if args.seed is not None:
        random.seed(args.seed)

# إعداد اللوج
    if args.log:
        logging.basicConfig(filename=args.log, level=logging.DEBUG, format=LOG_FORMAT)
    else:
        logging.basicConfig(level=logging.INFO if args.verbose else logging.WARNING, format=LOG_FORMAT)

    logger = logging.getLogger("simulator")
    logger.info("Starting simulation")
    logger.info(f"Scenario={args.scenario} duration={args.duration}s interval={args.interval}s
```

```
csv={args.csv}")

# تشغيل الحلقة الأسنكرونية
    loop = asyncio.get_event_loop()
    try:
        summary = loop.run_until_complete(run_simulation(args.duration, args.interval,
args.scenario, args.csv, args.verbose, args.transport))
    finally:
# في بعض بيئات بايثون قد نحتاج لإغلاق الحلقة
        try:
            loop.close()
        except Exception:
            pass

# طباعة الملخص
    print("\n=== Simulation Summary ===")
    print(f"Total readings: {summary['total_readings']}")
    print(f"Alerts sent: {summary['alerts_sent']}")
    print("Counts by type:")
    for k, v in summary["counters"].items():
        print(f" - {k}: {v}")
    print(f"Max Vibration: {summary['max_vibration']:.2f}" if summary["max_vibration"] is not
None else "Max Vibration: N/A")
    print(f"Max Temperature: {summary['max_temperature']:.2f}" if summary["max_temperature"]
is not None else "Max Temperature: N/A")
    print(f"Max Acoustic: {summary['max_acoustic']:.2f}" if summary["max_acoustic"] is not None
else "Max Acoustic: N/A")
    print(f"Avg Vibration: {summary['avg_vibration']:.2f}" if summary["avg_vibration"] is not None
else "Avg Vibration: N/A")
    print(f"Avg Temperature: {summary['avg_temperature']:.2f}" if summary['avg_temperature'] is
not None else "Avg Temperature: N/A")
    print(f"Avg Acoustic: {summary['avg_acoustic']:.2f}" if summary['avg_acoustic'] is not None
else "Avg Acoustic: N/A")
    print("==========================\n")

if __name__ == "__main__":
    main()
```