ДОДАТОК А
DOCKER COMPOSE ДЛЯ СЕРВЕРНОГО СТЕКУ

Файл docker-compose.yml для розгортання Mosquitto, Home Assistant, InfluxDB і Grafana

```
version: '3.8'

services:
  mosquitto:
    image: eclipse-mosquitto:2.0
    container_name: mqtt_broker
    restart: always
    volumes:
      - ./mosquitto/config:/mosquitto/config:ro
      - ./mosquitto/data:/mosquitto/data
      - ./mosquitto/log:/mosquitto/log
    ports:
      - "8883:8883"
      - "1883:1883"
    networks:
      - ha_net

  homeassistant:
    image: ghcr.io/home-assistant/home-assistant:2024.12
    container_name: home_assistant
    restart: always
    volumes:
      - ./homeassistant/config:/config
    environment:
```

```
    - TZ=Europe/Kyiv
  ports:
    - "8123:8123"
  depends_on:
    - mosquitto
  networks:
    - ha_net

influxdb:
image: influxdb:2.7
container_name: influxdb
volumes:
  - /opt/iot_stack/influxdb2:/var/lib/influxdb2
ports:
  - "8086:8086"
environment:
  - INFLUXDB_ADMIN_USER=admin
  - INFLUXDB_ADMIN_PASSWORD=secret_password
  - INFLUXDB_ORG=iot_org
  - INFLUXDB_BUCKET=iot_monitoring
  - INFLUXDB_RETENTION=2160h
  - INFLUXDB_INIT_MODE=setup
  - INFLUXDB_INIT_USERNAME=ha_user
  - INFLUXDB_INIT_PASSWORD=ha_password
  - INFLUXDB_INIT_TOKEN=ha_token_example
  - INFLUXDB_INIT_BUCKET=iot_monitoring
  - INFLUXDB_INIT_ORG=iot_org
  networks:
    - ha_net
```

```
  grafana:
    image: grafana/grafana:10.3.0
    container_name: grafana
    restart: always
    volumes:
      - ./grafana/data:/var/lib/grafana
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD=adminpass
      - GF_SECURITY_ALLOW_EMBEDDING=true
    ports:
      - "3000:3000"
    depends_on:
      - influxdb
    networks:
      - ha_net

networks:
  ha_net:
    driver: bridge
```

ДОДАТОК Б
КОД СКЕТЧУ ДЛЯ ESP32

Код скетчу IoT_Lux_Node.ino для ESP32

```
#include <WiFi.h>
#include <Wire.h>
#include <BH1750.h>
#include <FastLED.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <ArduinoOTA.h>
#include <SPIFFS.h>

#define WIFI_SSID       "IoT_AP"
#define WIFI_PASSWORD   "iot_pass"

#define MQTT_HOST       "192.168.10.10"
#define MQTT_PORT       8883
#define MQTT_USER       "esp32"
#define MQTT_PASS       "esp32_pass"

#define CA_CERT_PATH    "/ca.crt"

#define LUX_TOPIC       "livingroom/esp32_01/lux"
#define STATUS_TOPIC    "livingroom/esp32_01/status"

#define BH1750_SDA_PIN  32
#define BH1750_SCL_PIN  33

#define LED_PIN         25
```

```
#define LED_COUNT       64
#define LED_BRIGHTNESS  50

#define DEEP_SLEEP_SEC  30

BH1750 lightMeter;
CRGB leds[LED_COUNT];

WiFiClientSecure   tlsClient;
PubSubClient       mqttClient(tlsClient);

void setupWiFi();
void setupOTA();
void connectMQTT();
void publishLux();
void indicate(uint8_t r, uint8_t g, uint8_t b);

void setup() {
  Serial.begin(115200);
  delay(100);
  Serial.println("\n[BOOT] Lux-node starting…");

  if (!SPIFFS.begin(true)) {
    Serial.println("[ERROR] SPIFFS mount failed");
  }
  File caFile = SPIFFS.open(CA_CERT_PATH, "r");
  if (!caFile) {
    Serial.println("[ERROR] CA file not found");
  } else {
    tlsClient.setCACert(reinterpret_cast<const char*>(caFile.readString().c_str()));
```

```
  Serial.println("[INFO ] CA certificate loaded");
 }

 Wire.begin(BH1750_SDA_PIN, BH1750_SCL_PIN);
 if (lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE)) {
  Serial.println("[INFO ] BH1750 initialised");
 } else {
  Serial.println("[ERROR] BH1750 init failed");
 }

 FastLED.addLeds<WS2812B, LED_PIN, GRB>(leds, LED_COUNT);
 FastLED.setBrightness(LED_BRIGHTNESS);
 indicate(0, 0, 0);

 setupWiFi();
 setupOTA();

 mqttClient.setServer(MQTT_HOST, MQTT_PORT);
 connectMQTT();
}

void loop() {
 ArduinoOTA.handle();

 if (!mqttClient.connected()) {
  indicate(255, 255, 0);
  connectMQTT();
 }
 mqttClient.loop();
```

```
  publishLux();

  Serial.printf("[INFO ] Sleep %d s\n", DEEP_SLEEP_SEC);
  esp_sleep_enable_timer_wakeup((uint64_t)DEEP_SLEEP_SEC *
1'000'000ULL);
  delay(100);
  esp_deep_sleep_start();
}

void setupWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  indicate(0, 0, 128);
  Serial.printf("[INFO ] Connecting to Wi-Fi: %s\n", WIFI_SSID);

  uint8_t retries = 0;
  while (WiFi.status() != WL_CONNECTED && retries < 20) {
    delay(500);
    retries++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.printf("[INFO ] Wi-Fi OK, IP: %s\n", WiFi.localIP().toString().c_str());
    indicate(0, 255, 0);
  } else {
    Serial.println("[ERROR] Wi-Fi fail");
    indicate(255, 0, 0);
  }
}
```

```
void setupOTA() {
  ArduinoOTA.setHostname("esp32-lux-node");
  ArduinoOTA.setPassword("ota_password");
  ArduinoOTA.begin();
  Serial.println("[INFO ] OTA ready");
}

void connectMQTT() {
  Serial.printf("[INFO ] MQTT connect to %s:%d\n", MQTT_HOST,
MQTT_PORT);
  while (!mqttClient.connected()) {
    if (mqttClient.connect("esp32_lux_node", MQTT_USER, MQTT_PASS)) {
      Serial.println("[INFO ] MQTT connected");
      mqttClient.publish(STATUS_TOPIC, "online");
      indicate(0, 255, 0);
    } else {
      Serial.printf("[WARN ] MQTT rc=%d. Retry in 0.5 s\n", mqttClient.state());
      delay(500);
    }
  }
}

void publishLux() {
  float lux = lightMeter.readLightLevel();
  if (lux < 0) {
    Serial.println("[WARN ] BH1750 read error");
    return;
  }
  StaticJsonDocument<64> doc;
  doc["lux"] = lux;
```

```
  char buffer[64];
  size_t len = serializeJson(doc, buffer);

  if (mqttClient.publish(LUX_TOPIC, buffer, len)) {
    Serial.printf("[OK   ] Lux %.2f sent\n", lux);
    indicate(0, 128, 255);
    delay(100);
    indicate(0, 255, 0);
  } else {
    Serial.println("[FAIL ] Publish error");
    indicate(255, 0, 0);
  }
}

void indicate(uint8_t r, uint8_t g, uint8_t b) {
  for (uint8_t i = 0; i < LED_COUNT; i++) leds[i] = CRGB(r, g, b);
  FastLED.show();
}
```

ДОДАТОК В
FLUX-ЗАПИТ ДЛЯ ПАНЕЛІ У GRAFANA

Flux-запит для панелі «Лампа та освітленість» у Grafana

```
timeRange = (tables=<-) =>
  tables
    |> range(start: v.timeRangeStart, stop: v.timeRangeStop)

lamp_raw =
  from(bucket: "iot")
    |> timeRange()
    |> filter(fn: (r) =>
      r._measurement == "state" and
      r.entity_id == "umnaia_lampa_1_kabinet" and
      (r._field == "state" or r._field == "brightness")
    )
    |> pivot(
      rowKey: ["_time"],
      columnKey: ["_field"],
      valueColumn: "_value"
    )

lamp_on_off =
  lamp_raw
    |> map(fn: (r) => ({
      _time: r._time,
      _field: "on_off",
      _value: if r.state == "on" then 1.0 else 0.0
```

```
    }))
    |> aggregateWindow(
        every: v.windowPeriod,
        fn: last,
        createEmpty: true
      )
    |> fill(usePrevious: true)
    |> fill(value: 0.0)

lamp_brightness =
  lamp_raw
    |> map(fn: (r) => ({
      _time: r._time,
      _field: "brightness",
      _value: if r.state == "on" then float(v: r.brightness) else 0.0
    }))
    |> aggregateWindow(
        every: v.windowPeriod,
        fn: last,
        createEmpty: true
      )
    |> fill(usePrevious: true)
    |> fill(value: 0.0)

lamp_combined =
  union(tables: [lamp_on_off, lamp_brightness])
    |> pivot(
        rowKey: ["_time"],
        columnKey: ["_field"],
        valueColumn: "_value"
```

```
      )

lux =
  from(bucket: "iot")
    |> timeRange()
    |> filter(fn: (r) =>
      r._measurement == "lx" and
      r._field == "value" and
      r.domain == "sensor" and
      r.entity_id == "livingroom_lux"
    )
    |> aggregateWindow(
        every: v.windowPeriod,
        fn: mean,
        createEmpty: true
      )
    |> fill(usePrevious: true)
    |> map(fn: (r) => ({
      _time: r._time,
      lux: r._value
    }))
    |> keep(columns: ["_time", "lux"])

join(
  tables: {lamp: lamp_combined, lux: lux},
  on: ["_time"]
)
  |> yield(name: "lamp_and_lux")
```

ДОДАТОК Г
ДЕМОНСТРАЦІЙНИЙ МАТЕРІАЛ

Формат	Позиція.	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кільк.	Примітки
			Текстові документи		
А4	1	ГЮІК.411710.031 ПЗ	Пояснювальна записка	95	с.
			Інші документи		
	2		Демонстраційний матеріал		с.

					ГЮІК.411710.031 ВД
Змін.	Арк	Номер докум.	Підпис	Дата	
Розроб.		Прудніков Д.Д.			Розробка розподіленої IoT-системи для моніторингу та автоматичного управління параметрами розумного будинку
Перевір.		Янушкевич Д.А.			
Н.контр.		Демська Н.П.			
Затв.		Невлюдов І.Ш.			

Літера	Арку	Аркуш
Н	1	1

Кафедра КІТАР ХНУРЕ