ДОДАТОК А

Демонстраційний матеріал

ДОДАТОК Б

Програмний код серверу

```
#include <iostream>
#include <thread>
#include <chrono>
#include <cmath>
#include <string>
#include <mutex>
#include <deque>
#include <fstream>
#include <nlohmann/json.hpp>

#include "config.hpp"
#include "mqtt_client.hpp"

using json = nlohmann::json;

struct Pose {
    double x = 0;
    double y = 0;
    double theta = 0;
};

enum MissionStatus { PENDING, RUNNING, COMPLETED, FAILED };

struct Mission {
    std::string id;
    double px, py;
    double dx, dy;
    MissionStatus status = PENDING;
```

```
};

std::deque<Mission> missionQueue;
std::mutex queueMutex;

Pose pose;
std::mutex poseMutex;

bool inField(double x, double y) {
    return (x >= -5 && x <= 5 && y >= -5 && y <= 5);
}

double normalize(double a) {
    const double PI = 3.1415926535;
    while (a > PI) a -= 2 * PI;
    while (a < -PI) a += 2 * PI;
    return a;
}

bool goToPoint(MQTTClientWrapper& mqtt, double gx, double gy, const
std::string& label) {
    const double Kv = 0.35;
    const double Kw = 1;
    const double max_v = 0.3;
    const double max_w = 1.0;
    const double tol = 0.25;

    while (true) {
        Pose p;
        {
```

```
    std::lock_guard<std::mutex> lk(poseMutex);
    p = pose;
}

double dx = gx - p.x;
double dy = gy - p.y;
double dist = sqrt(dx * dx + dy * dy);

if (dist < tol) {
    mqtt.publish(config::TOPIC_CMD, "{\"linear\":0,\"angular\":0}");
    return true;
}

double target = atan2(dy, dx);
double err = normalize(target - p.theta);

double v = Kv * dist;
double w = Kw * err;

if (fabs(err) > 0.5)
    v = 0.05;

v = std::clamp(v, -max_v, max_v);
w = std::clamp(w, -max_w, max_w);

json cmd;
cmd["linear"] = v;
cmd["angular"] = w;
auto t_send = std::chrono::steady_clock::now();
std::cout << "[TIME] CMD sent at "
```

```
            << std::chrono::duration_cast<std::chrono::milliseconds>(
               t_send.time_since_epoch()).count()
            << " ms\n";
        auto t_back = std::chrono::steady_clock::now();
        auto rtt = std::chrono::duration_cast<std::chrono::milliseconds>(t_back -
t_send).count();
        std::cout << "[RTT] = " << rtt << " ms\n";
        mqtt.publish(config::TOPIC_CMD, cmd.dump());

        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }
}

int main() {
    std::cout << "[SERVER] Mission server online\n";

    MQTTClientWrapper mqtt(config::MQTT_BROKER_ADDRESS,
"mission_server");
    mqtt.connect();
    mqtt.subscribe(config::TOPIC_MISSION);
    mqtt.subscribe(config::TOPIC_STATE);

    mqtt.setCallback([&](const std::string& topic, const std::string& msg) {
        if (topic == config::TOPIC_STATE) {
            auto j = json::parse(msg);
            std::lock_guard<std::mutex> lock(poseMutex);
            pose.x = j["x"];
            pose.y = j["y"];
            pose.theta = j["theta"];
        }
```

```
    if (topic == config::TOPIC_MISSION) {
        auto j = json::parse(msg);
        Mission m;
        m.id = j["id"];
        m.px = j["pickup_x"];
        m.py = j["pickup_y"];
        m.dx = j["drop_x"];
        m.dy = j["drop_y"];

        if (!inField(m.px, m.py) || !inField(m.dx, m.dy)) {
            std::cout << "[SERVER][ERROR] Mission rejected — out of field!\n";
            return;
        }

        std::lock_guard<std::mutex> ql(queueMutex);
        missionQueue.push_back(m);

        std::cout << "[SERVER] Mission queued: " << m.id << "\n";
    }
    });

while (true) {
    Mission cur;

    {
        std::lock_guard<std::mutex> lock(queueMutex);
        if (missionQueue.empty()) {
            std::this_thread::sleep_for(std::chrono::milliseconds(200));
            continue;
```

```
            }
            cur = missionQueue.front();
            missionQueue.pop_front();
        }

        cur.status = RUNNING;
        std::cout << "[SERVER] Starting mission: " << cur.id << "\n";

        if (!goToPoint(mqtt, cur.px, cur.py, "PICKUP")) {
            cur.status = FAILED;
            continue;
        }

        mqtt.publish(config::TOPIC_CMD, "{\"linear\":0,\"angular\":0}");
        std::this_thread::sleep_for(std::chrono::seconds(3));

        if (!goToPoint(mqtt, cur.dx, cur.dy, "DROP")) {
            cur.status = FAILED;
            continue;
        }

        cur.status = COMPLETED;
        std::cout << "[SERVER] Mission done: " << cur.id << "\n";
    }

    return 0;
}
```

Формат	Позиція.	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кільк.	Примітки
			Текстові документи		
А4	1	ГЮІК 503320.005 ПЗ	Пояснювальна записка	73	с.
			Інші документи		
А4	2		Демонстраційний матеріал	14	с.
А4			Додаток Б програмний код серверу	1	с.

					ГЮІК 503320.005 ВД
Змін.	Арк	Номер докум.	Підпис	Дата	
Розроб.		Чеснаков В.А.			Розроблення програмного забезпечення для керування мобільною платформою на базі мережевих технологій у складській логістиці Відомість кваліфікаційної роботи
Переві		Артюх Р.В.			
Н.конт		Теслюк С.І			
Затв.		Невлюдов І.Ш.			

Літера	Арку	Аркуш
Н	1	1

Кафедра КІТАР ХНУРЕ