ДОДАТОК А

Текст програми

```
# GCC support can be specified at major, minor, or micro version
# (e.g. 8, 8.2 or 8.2.0).
# See https://hub.docker.com/r/library/gcc/ for all supported GCC
# tags from Docker Hub.
# See https://docs.docker.com/samples/library/gcc/ for more on how to use this image
FROM gcc:latest as robotinoserverenvironment

RUN apt-get update && apt-get -y install cmake python3 python3-pip mingw-w64
#sudo update-alternatives --config x86_64-w64-mingw32-g++
#sudo update-alternatives --config x86_64-w64-mingw32-gcc
RUN pip3 install conan

RUN conan remote add public-conan https://api.bintray.com/conan/bincrafters/public-conan
RUN conan remote add stiffstream https://api.bintray.com/conan/stiffstream/public

FROM robotinoserverenvironment AS robotinoserverbuilder

# These commands copy your files into the specified directory in the image
# and set that as the working location
COPY . /usr/src/RobotinoServer
WORKDIR /usr/build/RobotinoServer
```

```
# This command compiles your app using GCC, adjust for your source code
RUN cmake '/usr/src/RobotinoServer' -DCMAKE_BUILD_TYPE=Release
RUN cmake --build .

# This command runs your application, comment out this line to compile only
CMD ["./bin/RobotinoServer"]

LABEL Name=RobotinoServer Version=0.0.1

cmake_minimum_required(VERSION 3.18)

project(RobotinoServer)

set(CMAKE_EXPORT_COMPILE_COMMANDS ON)

set(CMAKE_CXX_STANDARD 17)

set(CMAKE_ARCHIVE_OUTPUT_DIRECTORY
${CMAKE_BINARY_DIR}/lib)
set(CMAKE_LIBRARY_OUTPUT_DIRECTORY
${CMAKE_BINARY_DIR}/lib)
set(CMAKE_RUNTIME_OUTPUT_DIRECTORY
${CMAKE_BINARY_DIR}/bin)

# Download automatically, you can also just copy the conan.cmake file
if(NOT EXISTS "${CMAKE_BINARY_DIR}/conan.cmake")
   message(STATUS "Downloading conan.cmake from
https://github.com/conan-io/cmake-conan")
```

```
  file(DOWNLOAD "https://github.com/conan-io/cmake-
conan/raw/v0.15/conan.cmake"
      "${CMAKE_BINARY_DIR}/conan.cmake"
      TLS_VERIFY ON)
endif()

include(${CMAKE_BINARY_DIR}/conan.cmake)

conan_cmake_run(
  REQUIRES
      boost/1.74.0
      restinio/0.6.12@stiffstream/stable
      nlohmann_json/3.9.1
      spdlog/1.8.1
      fmt/7.1.2
  BASIC_SETUP
  BUILD missing
  )

file(GLOB_RECURSE HEADERS include/*.hpp)
file(GLOB_RECURSE SOURCES src/*.cpp)

add_executable(${PROJECT_NAME}
  ${HEADERS}
  ${SOURCES}
  )

target_include_directories(${PROJECT_NAME}
  PRIVATE
      include
```

```
    )
target_link_libraries(${PROJECT_NAME} ${CONAN_LIBS} -lstdc++fs)

#include <chrono>
#include <future>

#include <restinio/all.hpp>
#include <spdlog/spdlog.h>

#include "http/handlers/PingHandler.hpp"
#include "http/handlers/BuildHandler.hpp"
#include "http/handlers/AdapterPollHandler.hpp"

using namespace std::chrono_literals;

constexpr auto ServerAddress = "localhost";
constexpr auto HttpServerPort = 5080;
constexpr auto WebSocketServerPort = 5082;

int main()
{
    struct HttpTraits : public restinio::default_single_thread_traits_t {
        using request_handler_t = restinio::router::express_router_t<>;
    };

    spdlog::info("Starting HTTP server on '{}:{}'...", ServerAddress,
HttpServerPort);

    restinio::router::express_router_t<> router;
```

```
    http::handlers::ping::createRouter(router);
    http::handlers::build::createRouter(router);
    http::handlers::adapterpoll::createRouter(router);

    restinio::run(
        restinio::on_thread_pool<HttpTraits>(8)
        .port(HttpServerPort)
        .address(ServerAddress)
        .request_handler(std::move(router)));

    while (true)
        std::this_thread::sleep_for(100ms);
}

#include <iostream>
#include <vector>

#include <boost/beast/http.hpp>
#include <boost/beast/core.hpp>
#include <boost/beast/websocket.hpp>
#include <boost/asio/connect.hpp>
#include <boost/asio/ip/tcp.hpp>

#include <rec/robotino/api2/c/globals.h>
#include <rec/robotino/api2/c/Com.h>
#include <rec/robotino/api2/c/Camera.h>

#include "base64.hpp"
#include "shared.hpp"
#include <future>
```

```
namespace beast = boost::beast;         // from <boost/beast.hpp>
namespace http = beast::http;           // from <boost/beast/http.hpp>
namespace websocket = beast::websocket; // from <boost/beast/websocket.hpp>
namespace net = boost::asio;            // from <boost/asio.hpp>
using tcp = boost::asio::ip::tcp;       // from <boost/asio/ip/tcp.hpp>

namespace
{

void InitConnection()
{
   auto com = Com_construct();

        Com_setAddress( com, "127.0.0.1:12080" );

   if(Com_connect( com ) == FALSE )
        {
      std::cout << "Robotino connection error" << std::endl;
      exit(1);
        }

        auto camera = Camera_construct();
        Camera_setComId( camera, com );
   Camera_setCameraNumber(camera, 0);
   Camera_setStreaming(camera, TRUE);

   for (;true; std::this_thread::sleep_for(std::chrono::milliseconds(100)))
   {
      if (!Camera_grab(camera))
```

```
            continue;

        Camera_setFormat(camera, 640, 480);

        unsigned int width, height;
        Camera_imageSize(camera, &width, &height);

        std::vector<uint8_t> data;
        data.resize(3*width*height);

        Camera_getImage(camera, data.data(), data.size(), &width, &height);

        net::io_context ioc;

        tcp::resolver resolver(ioc);
        beast::tcp_stream stream(ioc);

        auto const results = resolver.resolve("127.0.0.1", "8080");

        stream.connect(results);

        http::request<http::string_body> req(http::verb::post, "/camera", 11);
        req.set(http::field::host, "127.0.0.1");
        req.set(http::field::user_agent, BOOST_BEAST_VERSION_STRING);
        req.body() = base64_encode(data.data(), data.size());

        http::write(stream, req);
    }
}
```

```
}

int main()
{
    std::cout << "Starting Robotino Client..." << std::endl;

    auto thread = std::async(std::launch::async, InitConnection);

    try
    {
        RobotinoMain();
    }
    catch (const std::exception& e)
    {
        std::cout << "Client exception: " << e.what() << std::endl;
        return 1;
    }

    return 0;
}
```

ДОДАТОК Б

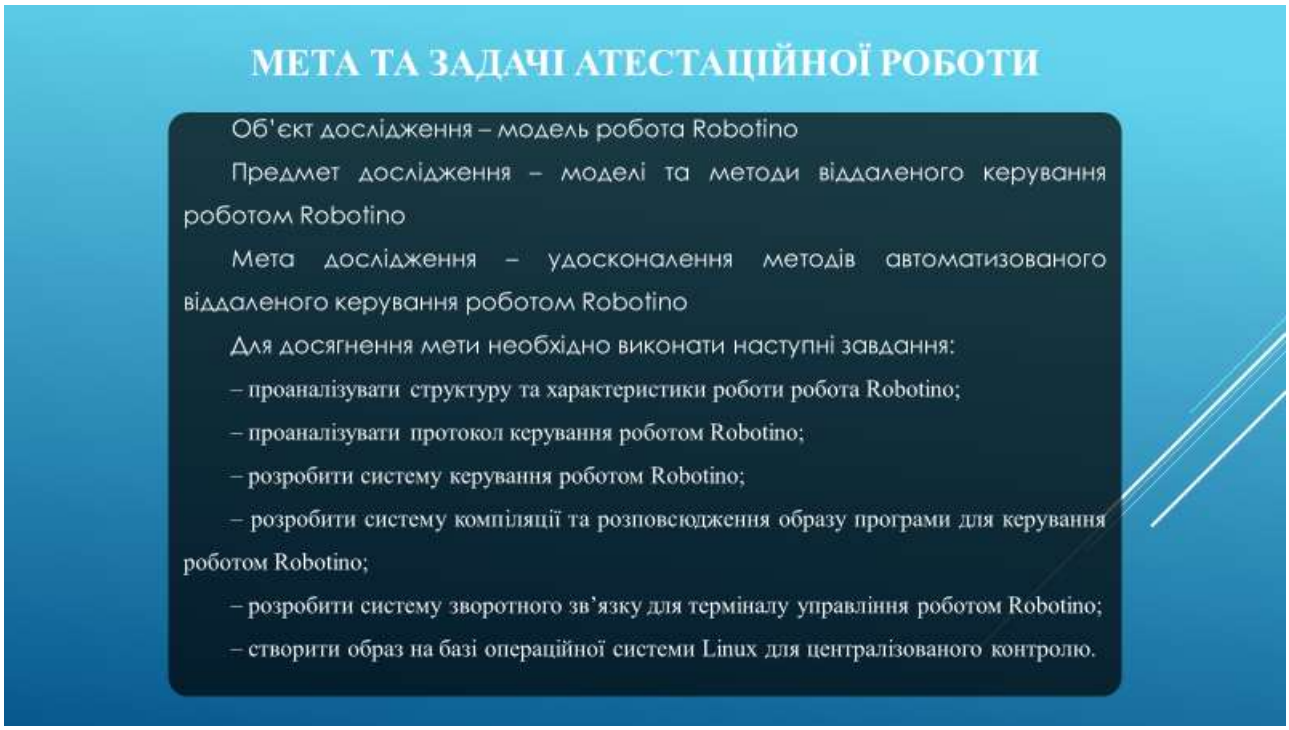
Демонстраційний матеріал

Слайд 1

ТИТУЛЬНИЙ ЛИСТ АТЕСТАЦІЙНОЇ РОБОТИ

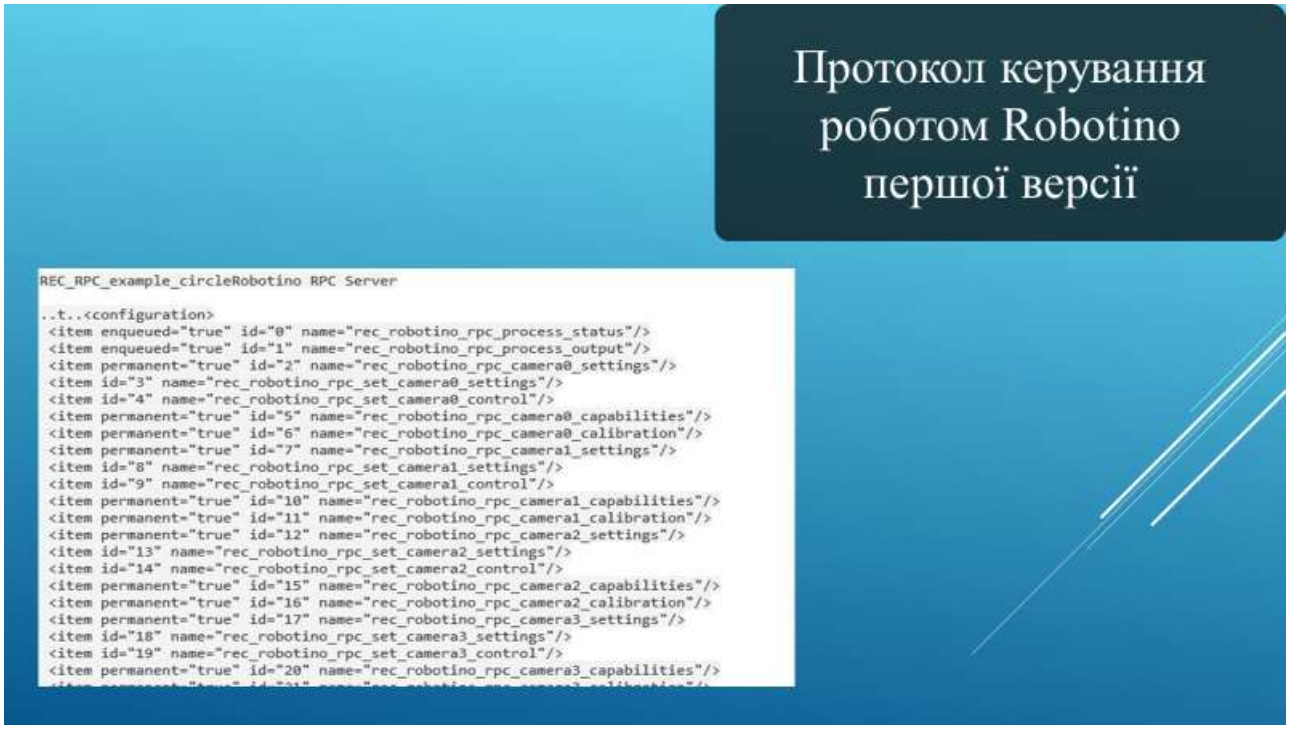
Слайд 2

МЕТА ТА ЗАДАЧІ АТЕСТАЦІЙНОЇ РОБОТИ



Слайд 3

ПРОТОКОЛ КЕРУВАННЯ РОБОТОМ ROBOTINO ПЕРШОЇ ВЕРСІЇ

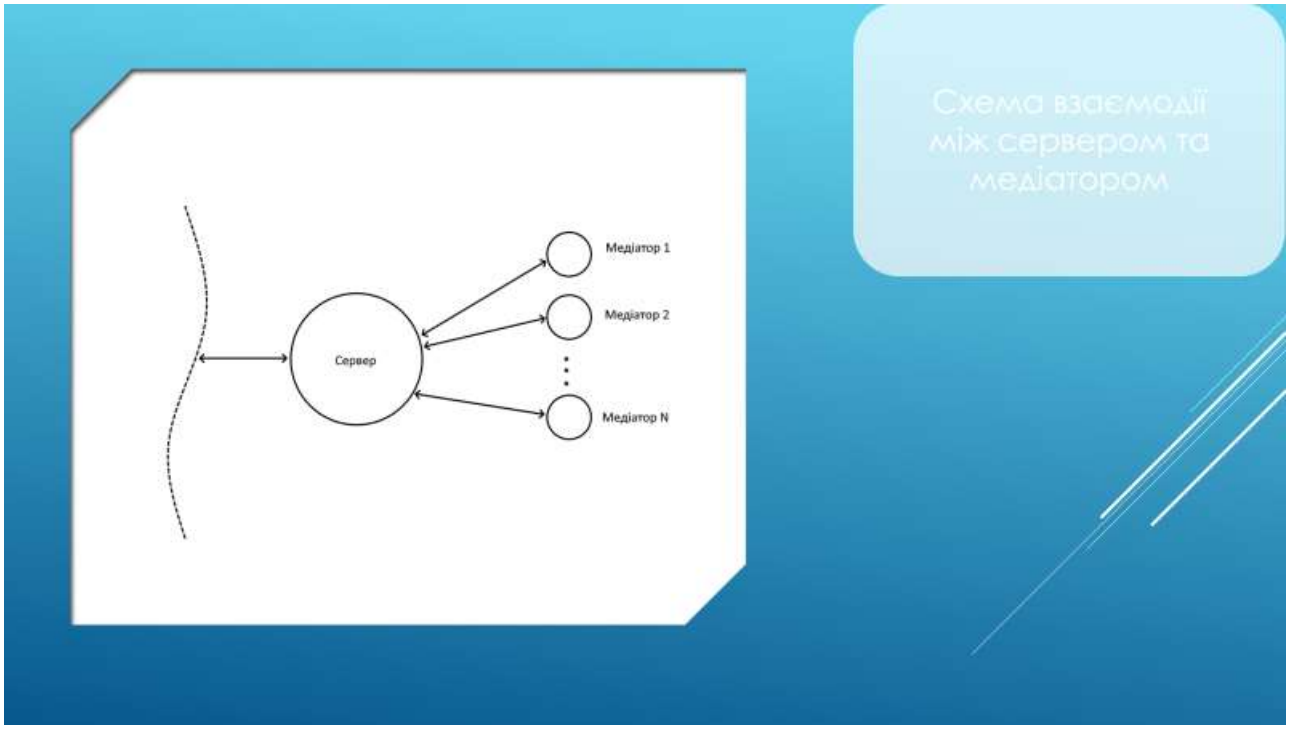


Слайд 4

ПРИКЛАД ТИПОВОЇ ПРОГРАМИ У СЕРЕДОВИЩІ ROBOTINO VIEW

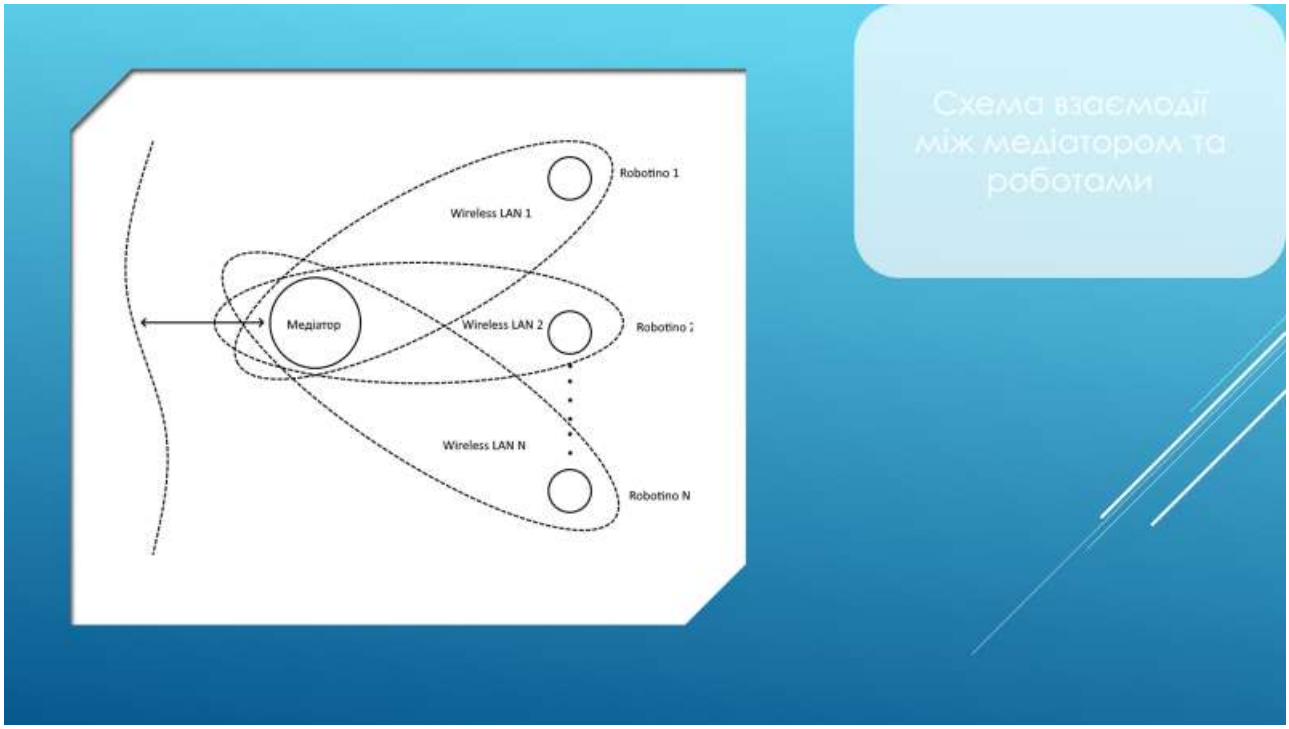
Слайд 5

СХЕМА ВЗАЄМОДІЇ МІЖ СЕРВЕРОМ ТА МЕДІАТОРОМ



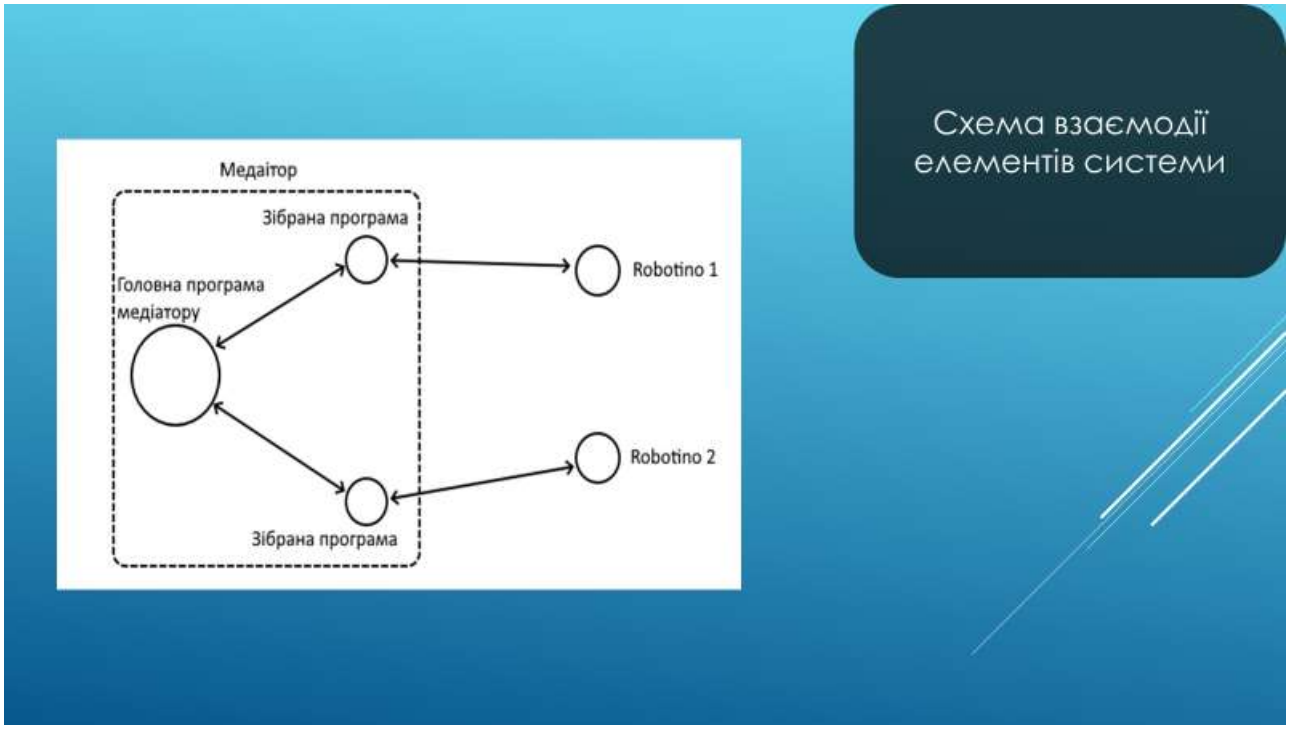
Слайд 6

СХЕМА ВЗАЄМОДІЇ МІЖ МЕДІАТОРОМ ТА РОБОТАМИ



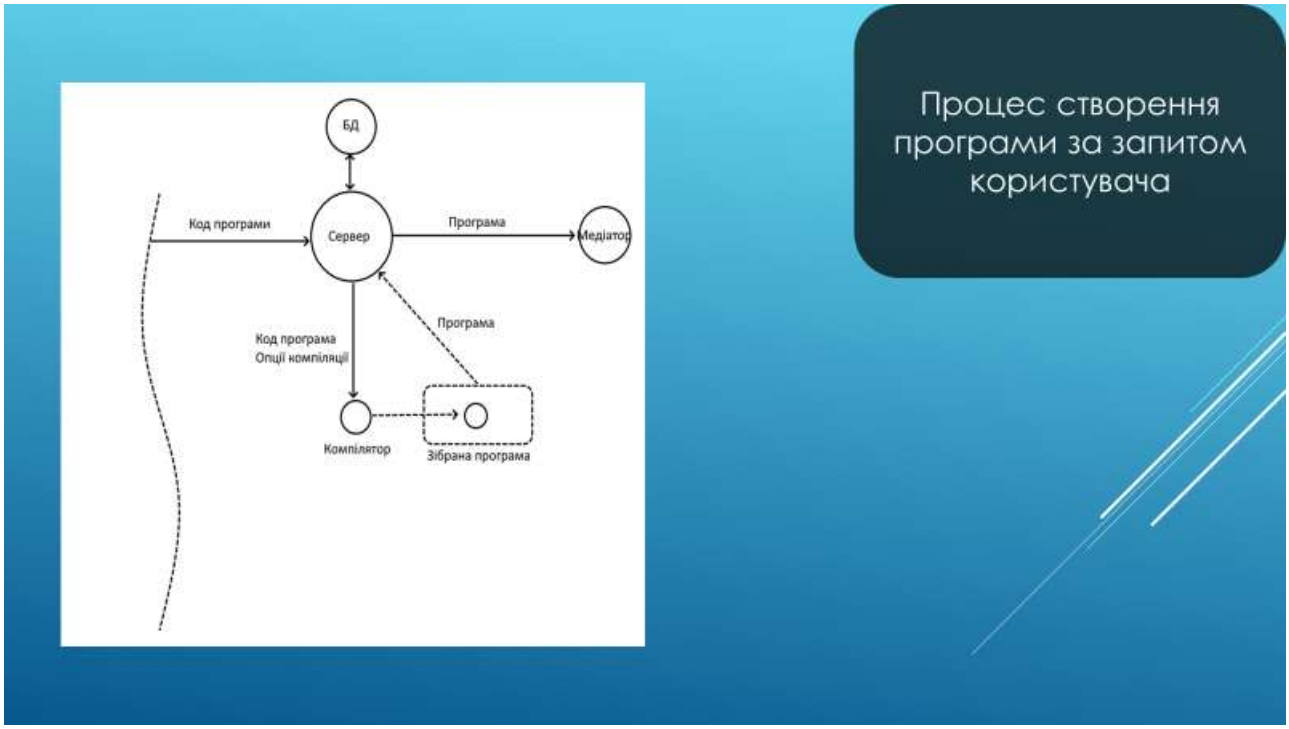
Слайд 7

СХЕМА ВЗАЄМОДІЇ ЕЛЕМЕНТІВ СИСТЕМИ



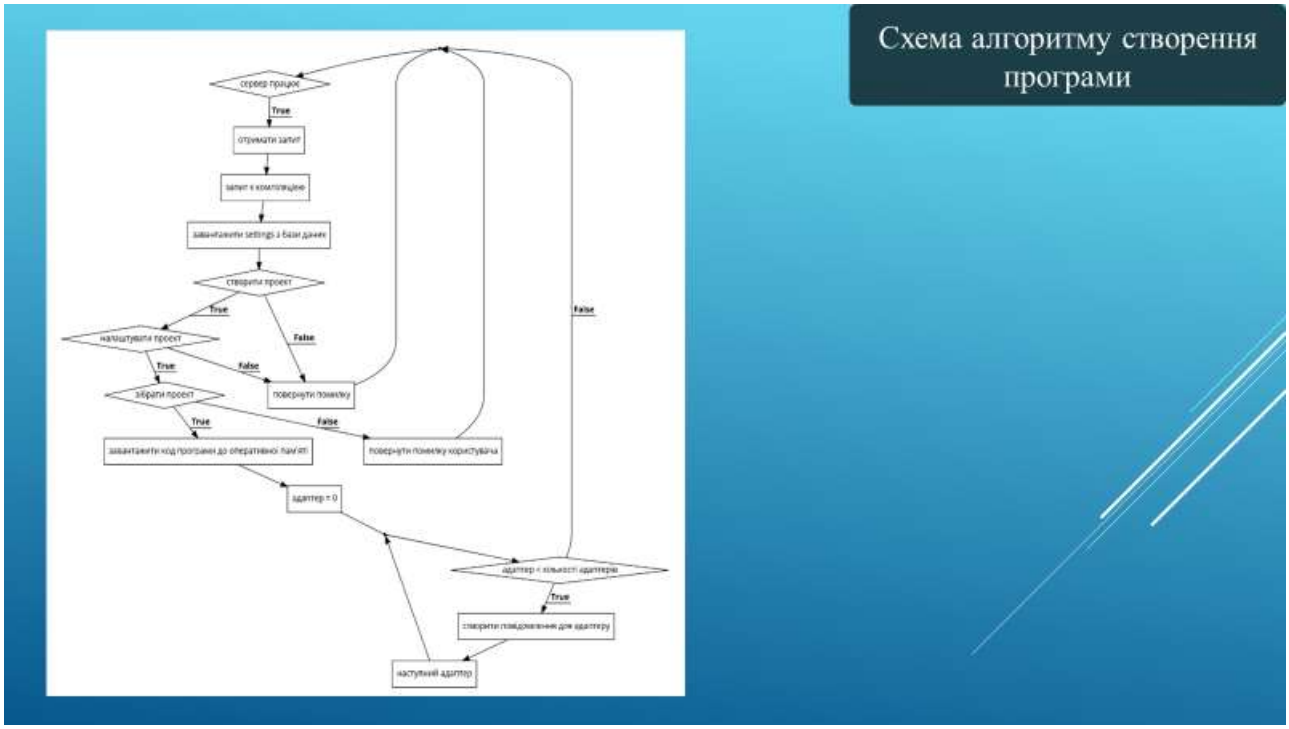
Слайд 8

ПРОЦЕС СТВОРЕННЯ ПРОГРАМИ ЗА ЗАПИТОМ КОРИСТУВАЧА



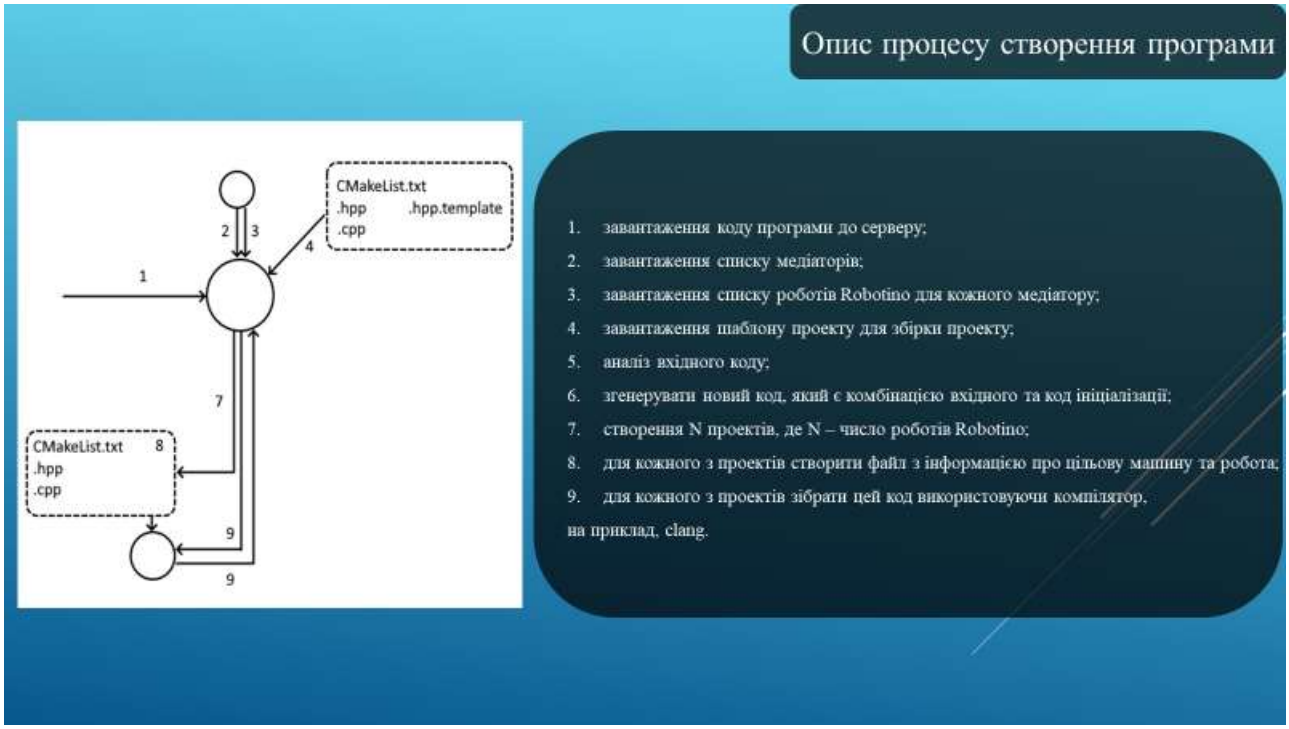
Слайд 9

СХЕМА АЛГОРИТМУ СТВОРЕННЯ ПРОГРАМИ



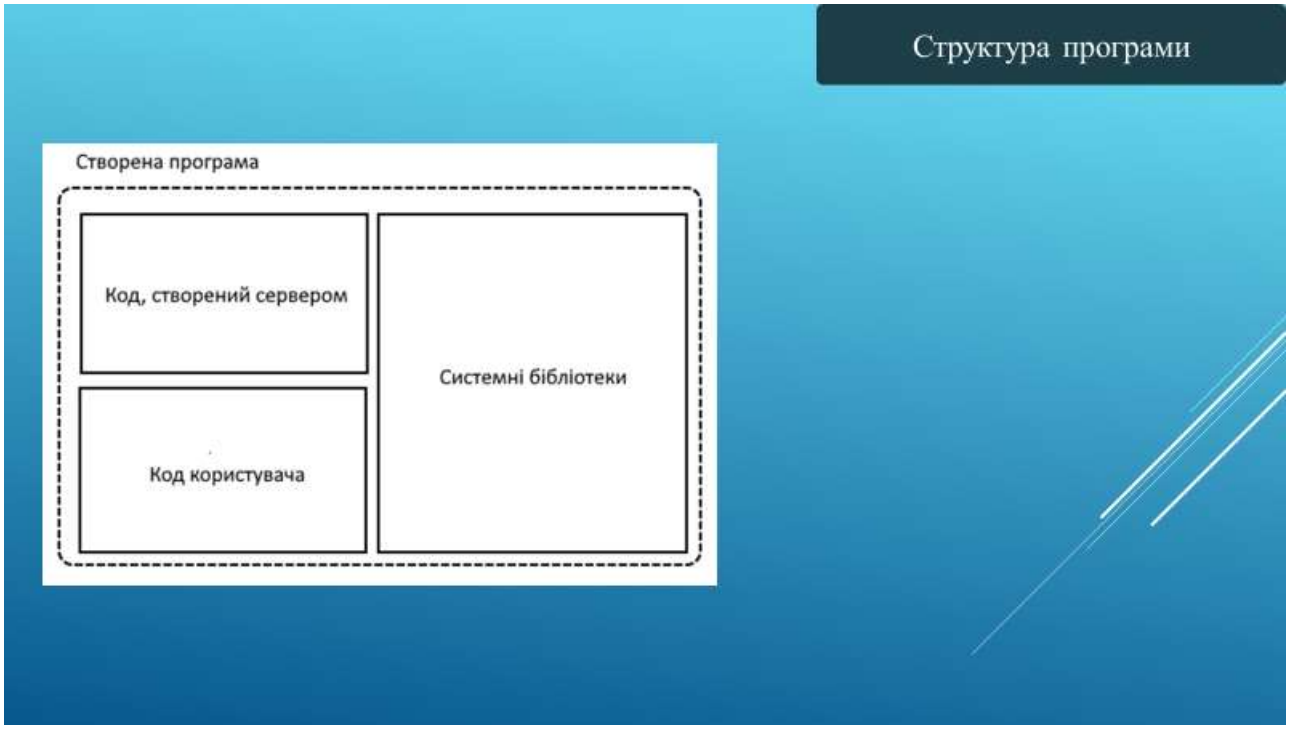
Слайд 10

ОПИС ПРОЦЕСУ СТВОРЕННЯ ПРОГРАМИ



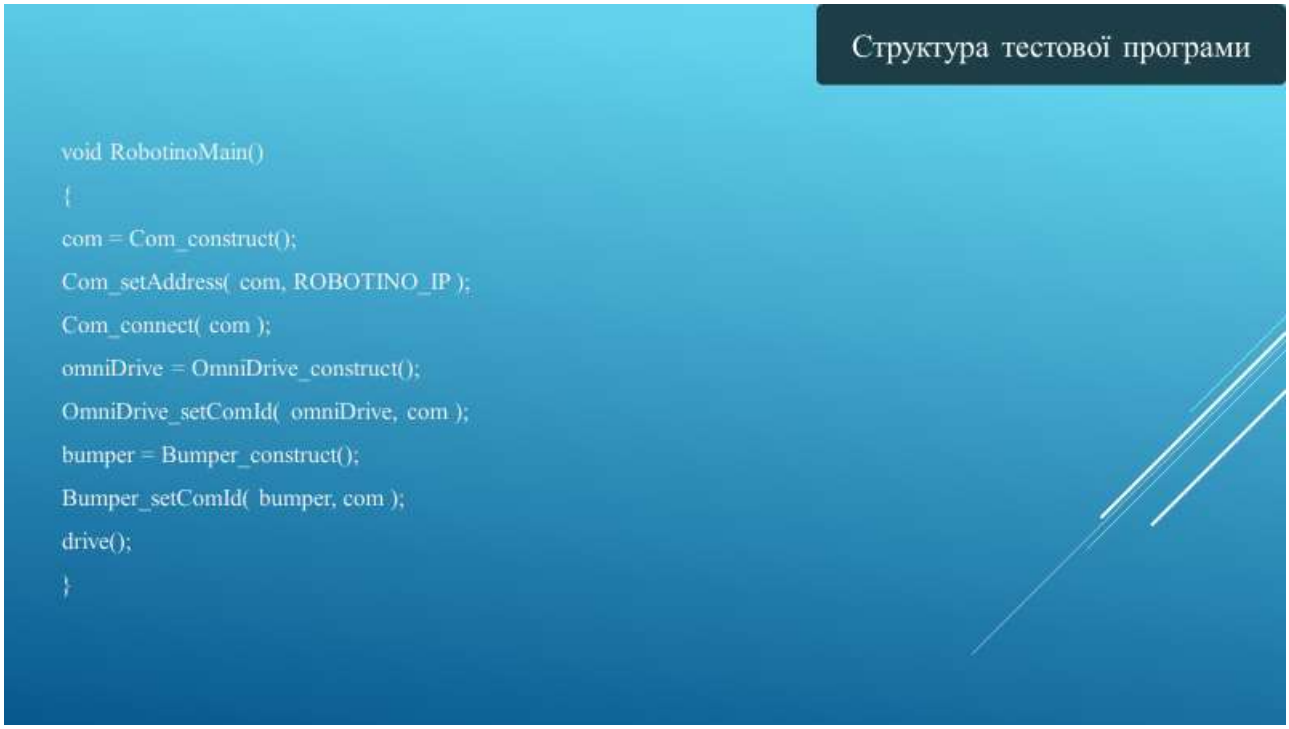
Слайд 11

СТРУКТУРА ПРОГРАМИ



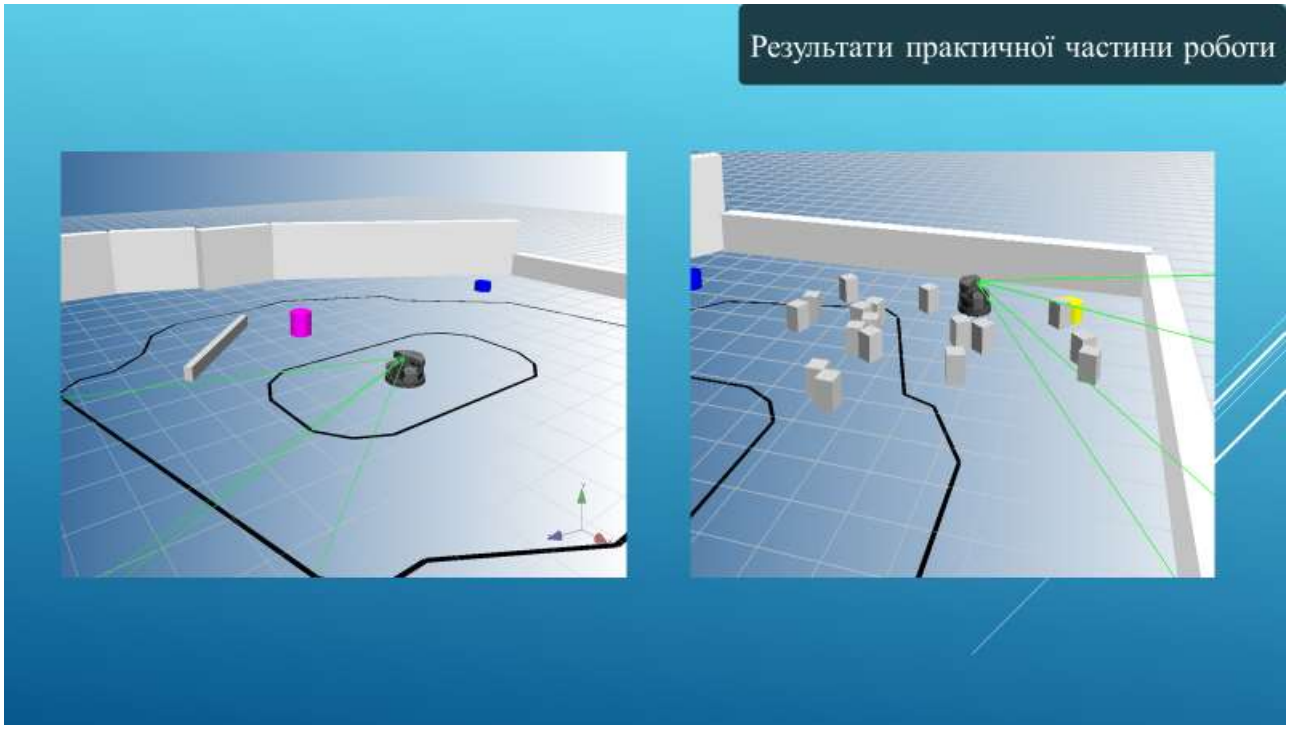
Слайд 12

СТРУКТУРА ТЕСТОВОЇ ПРОГРАМИ



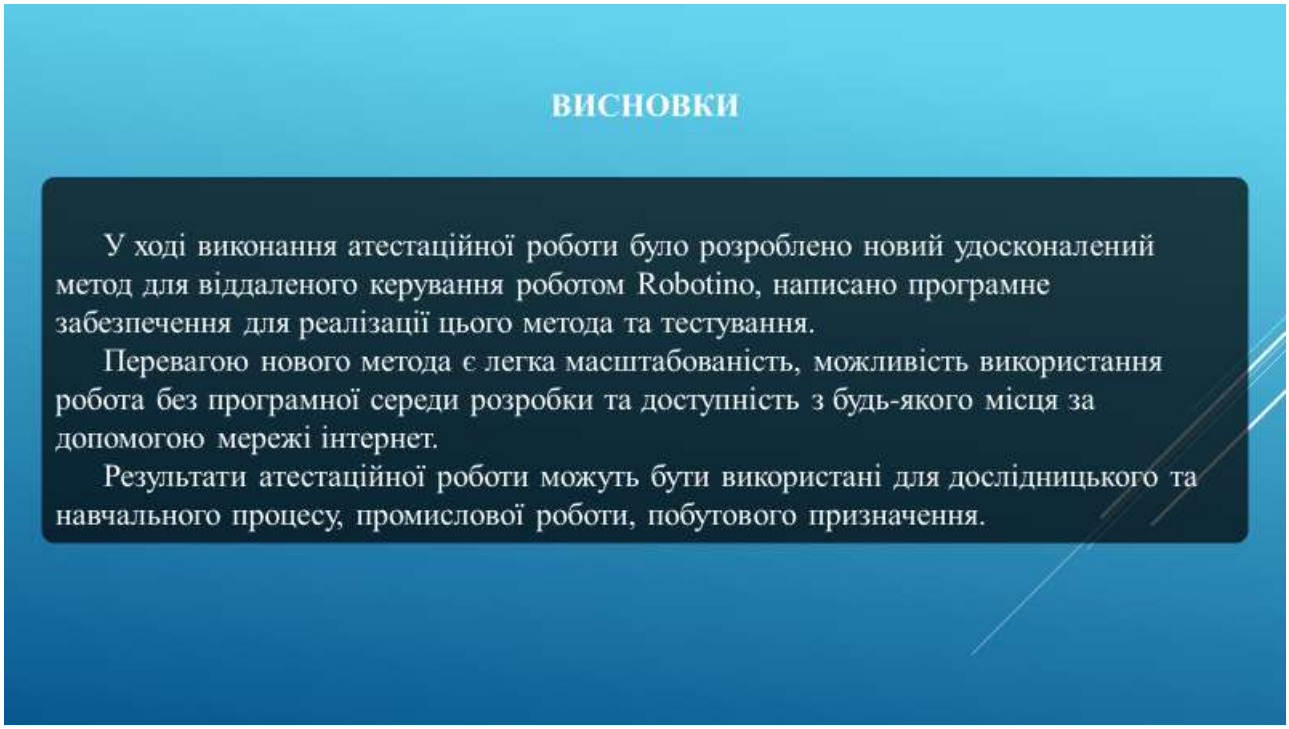
Слайд 13

РЕЗУЛЬТАТИ ПРАКТИЧНОЇ ЧАСТИНИ РОБОТИ



Слайд 14

ВИСНОВКИ



Слайд 15

ДЯКУЮ ЗА УВАГУ

ДОДАТОК В

Відомість атестаційної роботи

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1	ГЮИК.507160.006 ПЗ	Пояснювальна записка	92 с., ф. А4
		Додаткові матеріали	
2		Демонстраційний матеріал	у форматі
			*.ppt

Зм.	Арк.	№ докум.	Підпис	Дата	ГЮИК.507160.006 ВД			
Розробив		Крапивін В.С.			Розробка удосконалених засобів програмного керування мобільним роботом Festo Robotino Відомість атестаційної роботи	Літера	Аркуш	Аркушів
Перевір.		Сінотін А.М						1
						ХНУРЕ кафедра КІТАМ		
Н. контр.		Яшков І.О.						
Затв.		Невлюдов І.Ш.						