ДОДАТОК А

Програмний код для мікроконтролера ESP8622 (Arduino C++)

```
#include <ESP8622WiFi.h>
#include <ESP8622HTTPClient.h>
#include <ArduinoJson.h>

const char* ssid = "Wokwi-GUEST";
const String baseUrl = "http://a620-46-98-145-16.ngrok-free.app";
const int relayPin = D1;
bool locked = true;

void setup() {
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW);
  Serial.begin(115200);
  delay(1000);
  Serial.println("ESP почав працювати");

  WiFi.begin(ssid, "");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi з'єднано");
}

void loop() {
  if (WiFi.status() == WL_CONNECTED) {
```

```
    WiFiClient client;
    HTTPClient http;
    http.begin(client, baseUrl + "/status");
    int code = http.GET();

    if (code == 200) {
      String body = http.getString();
      StaticJsonDocument<200> doc;
      DeserializationError error = deserializeJson(doc, body);

      if (!error) {
        bool newState = doc["locked"];
        if (newState != locked) {
          locked = newState;
          digitalWrite(relayPin, locked ? HIGH : LOW);
          Serial.println(locked ? "Locked (on /lock)" : "Unlocked (on
/unlock)");
        }
      } else {
        Serial.println("JSON parse error");
      }
    } else {
      Serial.printf("HTTP error: %d\n", code);
    }

    http.end();
  } else {
    Serial.println("WiFi втрачено");
  }
  delay(2000);
}
```

ДОДАТОК Б

Програмний код для мікроконтролера ESP32 (Arduino C++) для перевірки роботоспособності схеми

```
#if defined(ESP8622)
#include <ESP8622WiFi.h>
#include <ESP8622HTTPClient.h>
#elif defined(ESP32)
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <WiFi.h>
#endif

const char* ssid = "Wokwi-GUEST";
//const String baseUrl = "http://a620-46-98-145-16.ngrok-free.app";
const int relayPin = 21;
bool locked = true;
String command = "";

void setup() {
  Serial.begin(115200);
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, HIGH);
  Serial.println("Система готова. Команди: /lock /unlock /status");
}

void loop() {
  if (Serial.available()) {
```

```
    command = Serial.readStringUntil('\n');
    command.trim();

    if (command == "/lock") {
      locked = true;
      digitalWrite(relayPin, HIGH);
      Serial.println("Замок заблоковано");
    } else if (command == "/unlock") {
      locked = false;
      digitalWrite(relayPin, LOW);
      Serial.println("Замок відчинено");
    } else if (command == "/status") {
      Serial.println(locked ? "Статус: заблоковано" : "Статус:
відчинено");
    } else {
      Serial.println("Невідома команда");
    }
  }
}
```

ДОДАТОК В

Лістинг коду мобільного застосунку Flutter

```
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';

void main() {
  runApp(LockApp());
}

class LockApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Smart Lock Controller',
      home: SplashScreen(),
    );
  }
}

// Splash перевіряє, чи є активний користувач
class SplashScreen extends StatefulWidget {
  @override
  _SplashScreenState createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen> {
```

```
  @override
  void initState() {
    super.initState();
    checkLoginStatus();
  }

  void checkLoginStatus() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    String? activeUser = prefs.getString('activeUser');
    if (activeUser != null) {
      Navigator.pushReplacement(context,
          MaterialPageRoute(builder: (_) => HomeScreen(username:
activeUser)));
    } else {
      Navigator.pushReplacement(
          context, MaterialPageRoute(builder: (_) => LoginScreen()));
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(body: Center(child: CircularProgressIndicator()));
  }
}

// Екран авторизації
class LoginScreen extends StatefulWidget {
  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
```

```
  final usernameController = TextEditingController();
  final passwordController = TextEditingController();

  void login() async {
    final username = usernameController.text.trim();
    final password = passwordController.text.trim();

    SharedPreferences prefs = await SharedPreferences.getInstance();
    final users = prefs.getStringList('users') ?? [];

    for (var entry in users) {
      final user = json.decode(entry);
      if (user['username'] == username && user['password'] == password) {
        await prefs.setString('activeUser', username);
        Navigator.pushReplacement(
          context,
          MaterialPageRoute(builder: (_) => HomeScreen(username:
username)),
        );
        return;
      }
    }

    ScaffoldMessenger.of(context)
        .showSnackBar(SnackBar(content: Text('Невірні дані входу')));
  }

  void goToRegister() {
    Navigator.push(
        context, MaterialPageRoute(builder: (_) => RegisterScreen()));
  }
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Вхід')),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            TextField(controller: usernameController, decoration:
InputDecoration(labelText: 'Ім’я користувача')),
            TextField(controller: passwordController, decoration:
InputDecoration(labelText: 'Пароль'), obscureText: true),
            SizedBox(height: 20),
            ElevatedButton(onPressed: login, child: Text('Увійти')),
            TextButton(onPressed: goToRegister, child:
Text('Реєстрація'))
          ],
        ),
      ),
    );
  }
}

// Екран реєстрації
class RegisterScreen extends StatefulWidget {
  @override
  _RegisterScreenState createState() => _RegisterScreenState();
}

class _RegisterScreenState extends State<RegisterScreen> {
  final usernameController = TextEditingController();
  final passwordController = TextEditingController();
```

```
  void register() async {
    final username = usernameController.text.trim();
    final password = passwordController.text.trim();

    SharedPreferences prefs = await SharedPreferences.getInstance();
    final users = prefs.getStringList('users') ?? [];

    for (var entry in users) {
      final user = json.decode(entry);
      if (user['username'] == username) {
        ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(content: Text('Користувач уже існує')));
        return;
      }
    }

    users.add(json.encode({'username': username, 'password': password}));
    await prefs.setStringList('users', users);
    await prefs.setString('activeUser', username);

    Navigator.pushReplacement(
        context, MaterialPageRoute(builder: (_) => HomeScreen(username:
username)));
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Реєстрація')),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
```

```
        child: Column(
          children: [
            TextField(controller: usernameController, decoration:
InputDecoration(labelText: 'Ім’я користувача')),
            TextField(controller: passwordController, decoration:
InputDecoration(labelText: 'Пароль'), obscureText: true),
            SizedBox(height: 20),
            ElevatedButton(onPressed: register, child:
Text('Зареєструватися')),
          ],
        ),
      ),
    );
  }
}

// Основний екран керування замком
class HomeScreen extends StatefulWidget {
  final String username;
  HomeScreen({required this.username});
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  String lockStatus = 'Невідомо';
  String ip = 'http://3b6e-46-174-68-144.ngrok-free.app';

  @override
  void initState() {
    super.initState();
    loadIP();
```

```
  }

  void loadIP() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    setState(() {
      ip = prefs.getString('deviceIP') ?? 'http://3b6e-46-174-68-
144.ngrok-free.app';
    });
  }

  Future<void> sendCommand(String command) async {
    try {
      final response = await http.post(Uri.parse('http://$ip/$command'));
      if (response.statusCode == 200) {
        setState(() {
          lockStatus = command == 'lock' ? 'Замкнено' : 'Відчинено';
        });
      } else {
        showError();
      }
    } catch (_) {
      showError();
    }
  }

  void showError() {
    ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Не вдалося з’єднатись з пристроєм')));
  }

  void logout() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
```

```
    await prefs.remove('activeUser');
    Navigator.pushReplacement(
        context, MaterialPageRoute(builder: (_) => LoginScreen()));
  }

  void goToSettings() {
    Navigator.push(
        context, MaterialPageRoute(builder: (_) => SettingsScreen()));
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Керування замком'),
        actions: [
          IconButton(onPressed: goToSettings, icon:
Icon(Icons.settings)),
          IconButton(onPressed: logout, icon: Icon(Icons.logout)),
        ],
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          children: [
            Text('Стан замка: $lockStatus', style: TextStyle(fontSize:
24)),
            SizedBox(height: 20),
            ElevatedButton(onPressed: () => sendCommand('unlock'), child:
Text('Відчинити')),
            ElevatedButton(onPressed: () => sendCommand('lock'), child:
Text('Замкнути')),
```

```
          ],
        ),
      ),
    );
  }
}

// Налаштування IP
class SettingsScreen extends StatefulWidget {
  @override
  _SettingsScreenState createState() => _SettingsScreenState();
}

class _SettingsScreenState extends State<SettingsScreen> {
  final ipController = TextEditingController();

  @override
  void initState() {
    super.initState();
    loadIP();
  }

  void loadIP() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    ipController.text = prefs.getString('deviceIP') ?? 'http://a620-46-
98-145-16.ngrok-free.app';
  }

  void saveIP() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    await prefs.setString('deviceIP', ipController.text.trim());
    Navigator.pop(context);
```

```
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Налаштування IP')),
      body: Padding(
        padding: EdgeInsets.all(16),
        child: Column(
          children: [
            TextField(controller: ipController, decoration:
InputDecoration(labelText: 'IP ESP8622')),
            SizedBox(height: 20),
            ElevatedButton(onPressed: saveIP, child: Text('Зберегти')),
          ],
        ),
      ),
    );
  }
}
```

ДОДАТОК Г

Демонстраційний матеріал

№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК. 504390.013 ПЗ	Пояснювальна записка	А4, 79 с.
		Додаткові матеріали	
2		Демонстраційний матеріал у вигляді презентації	А4, 12 с.

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК.504390.013 ВД		
Розроб.		Момот В.О.			Розроблення автоматизованої системи контролю замикання дверей із застосуванням IoT-технологій	Літера	Аркуш	Аркушів
Перевір.		Самойленко Г.Ю.				Н	1	1
Н.контр.		Демська Н.П.				Кафедра КІТАР ХНУРЕ		
Затв.		Невлюдов І.Ш.						