ДОДАТОК А

Лістинг основного коду системи орошення

```
#include <Wi-Fi.h>
#include <WebServer.h>
#include <EEPROM.h>
#include <ArduinoJson.h>
#include <time.h>

#define MOTOR_IN1 4
#define MOTOR_IN2 12
#define MOTOR_IN3 16
#define MOTOR_IN4 15

#define MOISTURE_SENSOR 13
#define VALVE_RELAY 14
#define HALL_SENSOR 2

const char* ssid = "ESP32_Irrigation";
const char* password = "12345678";

#define EEPROM_SIZE 512
#define ADMIN_PASSWORD "admin123"

struct Settings {
  int recommendedMoisture;
  int moistureBuffer;
  int checkInterval;
  char plantName[51];
  time_t lastWateringTime;
};

struct LogEntry {
  time_t timestamp;
  int moistureLevel;
  bool isWatering;
};

Settings settings;
WebServer server(80);
unsigned long lastCheckTime = 0;
int currentMoisture = 0;
bool isWatering = false;
bool isMoving = false;
int robotPosition = 0;
bool manualCheckRequested = false;
bool manualWateringRequested = false;
```

```
LogEntry moistureLog[10];
int logIndex = 0;
unsigned long lastSerialPrintTime = 0;

enum RobotState {
  IDLE,
  MOVING_TO_ZONE,
  CHECKING_MOISTURE,
  WATERING,
  RETURNING_HOME,
  ERROR
};

RobotState currentState = IDLE;
String errorMessage = "";

void loadSettings() {
  EEPROM.get(0, settings);

  if (settings.recommendedMoisture < 10 || settings.recommendedMoisture > 90) {
    settings.recommendedMoisture = 70;
    Serial.println("Default recommended moisture set: 70%");
  }

  if (settings.moistureBuffer < 0 || settings.moistureBuffer > 30) {
    settings.moistureBuffer = 10;
    Serial.println("Default moisture buffer set: 10%");
  }

  if (settings.checkInterval < 5 || settings.checkInterval > 1440) {
    settings.checkInterval = 15;
    Serial.println("Default check interval set: 15 minutes");
  }

  if (strlen(settings.plantName) == 0) {
    strcpy(settings.plantName, "Plant");
    Serial.println("Default plant name set: Plant");
  }

  if (settings.lastWateringTime == 0) {
    settings.lastWateringTime = 0;
    Serial.println("Last watering time not set");
  }
}

void saveSettings() {
  EEPROM.put(0, settings);
  if (EEPROM.commit()) {
    Serial.println("Settings successfully saved to EEPROM");
  } else {
    Serial.println("Error saving settings to EEPROM");
  }
}
```

```
int getMoistureLevel() {
  int moistureValue;
  if (digitalRead(MOISTURE_SENSOR) == LOW) {
    moistureValue = random(60, 100);
  } else {
    moistureValue = random(10, 60);
  }

  Serial.print("Measured moisture: ");
  Serial.print(moistureValue);
  Serial.println("%");

  return moistureValue;
}

void controlMotors(bool forward) {
  if (forward) {
    digitalWrite(MOTOR_IN1, HIGH);
    digitalWrite(MOTOR_IN2, LOW);
    digitalWrite(MOTOR_IN3, HIGH);
    digitalWrite(MOTOR_IN4, LOW);
    Serial.println("Motors: moving forward");
  } else {
    digitalWrite(MOTOR_IN1, LOW);
    digitalWrite(MOTOR_IN2, HIGH);
    digitalWrite(MOTOR_IN3, LOW);
    digitalWrite(MOTOR_IN4, HIGH);
    Serial.println("Motors: moving backward");
  }
}

void stopMotors() {
  digitalWrite(MOTOR_IN1, LOW);
  digitalWrite(MOTOR_IN2, LOW);
  digitalWrite(MOTOR_IN3, LOW);
  digitalWrite(MOTOR_IN4, LOW);
  Serial.println("Motors: stopped");
}

void controlValve(bool open) {
  if (open) {
    digitalWrite(VALVE_RELAY, HIGH);
    Serial.println("Valve: open");
  } else {
    digitalWrite(VALVE_RELAY, LOW);
    Serial.println("Valve: closed");
  }
}

bool checkHallSensor() {
  bool hallState = digitalRead(HALL_SENSOR) == LOW;
  if (hallState) {
```

```
    Serial.println("Hall sensor: magnet detected");
  }
  return hallState;
}

bool moveToZone() {
  currentState = MOVING_TO_ZONE;
  Serial.println("State: moving to zone");

  if (robotPosition != 0) {
    Serial.println("Robot not at home position, returning home");
    return returnHome();
  }

  controlMotors(true);
  isMoving = true;

  unsigned long startTime = millis();
  unsigned long timeout = 30000;

  Serial.println("Moving forward until magnet 2 detected (zone start)");

  while (millis() - startTime < timeout) {
    if (checkHallSensor()) {
      delay(500);

      if (robotPosition == 0) {
        robotPosition = 1;
        Serial.println("Magnet 2 detected (zone start)");
        stopMotors();
        isMoving = false;
        return true;
      } else if (robotPosition == 2) {
        robotPosition = 3;
        Serial.println("Magnet 4 detected (track end)");
        stopMotors();
        delay(1000);
        controlMotors(false);
      }
    }

    if (millis() - lastSerialPrintTime > 5000) {
      Serial.print("Moving to zone, elapsed time: ");
      Serial.print((millis() - startTime) / 1000);
      Serial.println(" sec");
      lastSerialPrintTime = millis();
    }

    delay(100);
  }

  stopMotors();
  isMoving = false;
```

```
  currentState = ERROR;
  errorMessage = "Timeout while moving to zone";
  Serial.println("Error: timeout while moving to zone");
  return false;
}

bool returnHome() {
  currentState = RETURNING_HOME;
  Serial.println("State: returning home");

  if (robotPosition == 0) {
    Serial.println("Robot already at home position");
    return true;
  }

  controlMotors(false);
  isMoving = true;

  unsigned long startTime = millis();
  unsigned long timeout = 30000;

  Serial.println("Moving backward until magnet 1 detected (home position)");

  while (millis() - startTime < timeout) {
    if (checkHallSensor()) {
      delay(500);

      if (robotPosition == 1) {
        robotPosition = 0;
        Serial.println("Magnet 1 detected (home position)");
        stopMotors();
        isMoving = false;
        return true;
      }
    }

    if (millis() - lastSerialPrintTime > 5000) {
      Serial.print("Returning home, elapsed time: ");
      Serial.print((millis() - startTime) / 1000);
      Serial.println(" sec");
      lastSerialPrintTime = millis();
    }

    delay(100);
  }

  stopMotors();
  isMoving = false;
  currentState = ERROR;
  errorMessage = "Timeout while returning home";
  Serial.println("Error: timeout while returning home");
  return false;
}
```

```
void moveWithinZone() {
  static bool movingForward = true;

  if (movingForward) {
    controlMotors(true);
    isMoving = true;

    if (checkHallSensor() && robotPosition == 1) {
      robotPosition = 2;
      Serial.println("Magnet 3 detected (zone end)");
      movingForward = false;
      stopMotors();
      delay(1000);
      controlMotors(false);
    }
  } else {
    controlMotors(false);
    isMoving = true;

    if (checkHallSensor() && robotPosition == 2) {
      robotPosition = 1;
      Serial.println("Magnet 2 detected (zone start)");
      movingForward = true;
      stopMotors();
      delay(1000);
      controlMotors(true);
    }
  }
}

void checkMoisture() {
  Serial.println("Starting moisture check");

  if (moveToZone()) {
    currentState = CHECKING_MOISTURE;
    Serial.println("State: checking moisture");

    currentMoisture = getMoistureLevel();

    moistureLog[logIndex].timestamp = time(NULL);
    moistureLog[logIndex].moistureLevel = currentMoisture;
    moistureLog[logIndex].isWatering = false;
    logIndex = (logIndex + 1) % 10;
    Serial.println("Log entry: moisture measurement");

    returnHome();

    manualCheckRequested = false;

    lastCheckTime = millis();

    currentState = IDLE;
```

```
    Serial.println("State: idle");
  } else {
    Serial.println("Failed to move to zone for moisture check");
  }
}

void waterZone() {
  Serial.println("Starting watering");

  if (moveToZone()) {
    currentState = WATERING;
    isWatering = true;
    Serial.println("State: watering");

    int targetMoisture = settings.recommendedMoisture + settings.moistureBuffer;
    Serial.print("Target moisture: ");
    Serial.print(targetMoisture);
    Serial.println("%");

    controlValve(true);

    int initialMoisture = getMoistureLevel();
    Serial.print("Initial moisture: ");
    Serial.print(initialMoisture);
    Serial.println("%");

    moistureLog[logIndex].timestamp = time(NULL);
    moistureLog[logIndex].moistureLevel = initialMoisture;
    moistureLog[logIndex].isWatering = true;
    logIndex = (logIndex + 1) % 10;
    Serial.println("Log entry: watering start");

    unsigned long startTime = millis();
    unsigned long timeout = 300000;

    Serial.println("Starting movement within zone for watering");

    while (currentMoisture < targetMoisture && millis() - startTime < timeout) {
      moveWithinZone();

      if (millis() % 5000 < 100) {
        currentMoisture = getMoistureLevel();
      }

      if (millis() - lastSerialPrintTime > 5000) {
        Serial.print("Watering, current moisture: ");
        Serial.print(currentMoisture);
        Serial.print("%, target: ");
        Serial.print(targetMoisture);
        Serial.print("%, elapsed time: ");
        Serial.print((millis() - startTime) / 1000);
        Serial.println(" sec");
        lastSerialPrintTime = millis();
```

```
      }

      delay(100);
    }

    controlValve(false);
    stopMotors();
    isMoving = false;
    isWatering = false;

    moistureLog[logIndex].timestamp = time(NULL);
    moistureLog[logIndex].moistureLevel = currentMoisture;
    moistureLog[logIndex].isWatering = false;
    logIndex = (logIndex + 1) % 10;
    Serial.println("Log entry: watering end");

    settings.lastWateringTime = time(NULL);
    saveSettings();

    returnHome();

    manualWateringRequested = false;

    currentState = IDLE;
    Serial.println("State: idle");
  } else {
    Serial.println("Failed to move to zone for watering");
  }
}

bool needWatering() {
  bool need = currentMoisture < settings.recommendedMoisture;
  if (need) {
    Serial.println("Watering required: current moisture below recommended");
  }
  return need;
}

void setup() {
  Serial.begin(115200);

  delay(3000);

  Serial.println("\n\n");
  Serial.println("=====================================");
  Serial.println("Initializing auto irrigation system...");
  Serial.println("=====================================");

  pinMode(MOTOR_IN1, OUTPUT);
  pinMode(MOTOR_IN2, OUTPUT);
  pinMode(MOTOR_IN3, OUTPUT);
  pinMode(MOTOR_IN4, OUTPUT);
  pinMode(MOISTURE_SENSOR, INPUT);
```

```
  pinMode(VALVE_RELAY, OUTPUT);
  pinMode(HALL_SENSOR, INPUT_PULLUP);

  Serial.println("Pins initialized");

  stopMotors();
  controlValve(false);

  EEPROM.begin(EEPROM_SIZE);
  Serial.println("EEPROM initialized");
  loadSettings();

  for (int i = 0; i < 10; i++) {
    moistureLog[i].timestamp = 0;
    moistureLog[i].moistureLevel = 0;
    moistureLog[i].isWatering = false;
  }
  Serial.println("Log initialized");

  Serial.print("Setting up Wi-Fi access point: ");
  Serial.println(ssid);
  Wi-Fi.softAP(ssid, password);
  IPAddress IP = Wi-Fi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);

  server.on("/", HTTP_GET, handleRoot);
  server.on("/check", HTTP_POST, handleCheck);
  server.on("/water", HTTP_POST, handleWater);
  server.on("/home", HTTP_POST, handleHome);
  server.on("/settings", HTTP_POST, handleSettings);
  server.onNotFound(handleNotFound);

  server.begin();
  Serial.println("HTTP server started");

  configTime(0, 0, "pool.ntp.org", "time.nist.gov");
  Serial.println("Time configuration completed");

  currentMoisture = getMoistureLevel();

  Serial.println("====================================");
  Serial.println("System ready");
  Serial.println("====================================");
}

void loop() {
  server.handleClient();

  if (currentState == IDLE) {
    unsigned long currentTime = millis();
    if ((currentTime - lastCheckTime > settings.checkInterval * 60000) ||
manualCheckRequested) {
```

```
      checkMoisture();
    }

    if (needWatering() || manualWateringRequested) {
      waterZone();
    }
  }

  if (currentState == WATERING && isMoving) {
    moveWithinZone();
  }

  if (millis() - lastSerialPrintTime > 30000) {
    Serial.println("\n--- System Status ---");
    Serial.print("State: ");
    switch (currentState) {
      case IDLE: Serial.println("Idle"); break;
      case MOVING_TO_ZONE: Serial.println("Moving to zone"); break;
      case CHECKING_MOISTURE: Serial.println("Checking moisture"); break;
      case WATERING: Serial.println("Watering"); break;
      case RETURNING_HOME: Serial.println("Returning home"); break;
      case ERROR: Serial.println("Error: " + errorMessage); break;
    }
    Serial.print("Current moisture: ");
    Serial.print(currentMoisture);
    Serial.println("%");
    Serial.print("Robot position: ");
    Serial.println(robotPosition);
    Serial.print("Free memory: ");
    Serial.print(ESP.getFreeHeap());
    Serial.println(" bytes");
    Serial.println("----------------------");

    lastSerialPrintTime = millis();
  }

  delay(100);
}
```

ДОДАТОК Б

Лістинг коду веб-інтерфейсу та обробників HTTP запитів

```
const char* htmlHeader = R"(
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Auto Irrigation System</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 20px;
            background-color: #f5f5f5;
        }
        .container {
            max-width: 800px;
            margin: 0 auto;
            background-color: white;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        h1 {
            color: #2c3e50;
            text-align: center;
        }
        .status-panel {
            background-color: #f9f9f9;
            padding: 15px;
            border-radius: 5px;
            margin-bottom: 20px;
        }
        .status-item {
            margin-bottom: 10px;
        }
        .status-label {
            font-weight: bold;
            display: inline-block;
            width: 200px;
        }
        .btn {
            background-color: #3498db;
            color: white;
```

```
            border: none;
            padding: 10px 15px;
            border-radius: 5px;
            cursor: pointer;
            margin-right: 10px;
            margin-bottom: 10px;
        }
        .btn:hover {
            background-color: #2980b9;
        }
        .btn-danger {
            background-color: #e74c3c;
        }
        .btn-danger:hover {
            background-color: #c0392b;
        }
        .form-group {
            margin-bottom: 15px;
        }
        label {
            display: block;
            margin-bottom: 5px;
            font-weight: bold;
        }
        input[type="text"],
        input[type="number"],
        input[type="password"] {
            width: 100%;
            padding: 8px;
            border: 1px solid #ddd;
            border-radius: 4px;
            box-sizing: border-box;
        }
        .log-table {
            width: 100%;
            border-collapse: collapse;
            margin-top: 20px;
        }
        .log-table th, .log-table td {
            border: 1px solid #ddd;
            padding: 8px;
            text-align: left;
        }
        .log-table th {
            background-color: #f2f2f2;
        }
        .alert {
            padding: 15px;
            background-color: #f8d7da;
            color: #721c24;
            border-radius: 5px;
            margin-bottom: 20px;
            display: none;
```

```
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Auto Irrigation System</h1>
)";

const char* htmlFooter = R"(
    </div>
    <script>
        function confirmAction(message, formId) {
            if (confirm(message)) {
                document.getElementById(formId).submit();
            }
        }

        function validateSettings() {
            var recommendedMoisture =
parseInt(document.getElementById('recommendedMoisture').value);
            var moistureBuffer =
parseInt(document.getElementById('moistureBuffer').value);

            if (moistureBuffer > 10) {
                var password = prompt('Password required for moisture buffer over
10%:');
                if (password) {
                    document.getElementById('adminPassword').value = password;
                } else {
                    return false;
                }
            }

            return confirm('Are you sure you want to change the settings?');
        }
    </script>
</body>
</html>
)";

void handleRoot() {
  Serial.println("Request: main page");

  String html = htmlHeader;

  html += "<div class='status-panel'>";
  html += "<h2>System Status</h2>";

  html += "<div class='status-item'><span class='status-label'>State:</span> ";
  switch (currentState) {
    case IDLE:
      html += "Idle";
      break;
```

```
    case MOVING_TO_ZONE:
      html += "Moving to zone";
      break;
    case CHECKING_MOISTURE:
      html += "Checking moisture";
      break;
    case WATERING:
      html += "Watering";
      break;
    case RETURNING_HOME:
      html += "Returning to home position";
      break;
    case ERROR:
      html += "Error: " + errorMessage;
      break;
  }
  html += "</div>";

  html += "<div class='status-item'><span class='status-label'>Current
moisture:</span> " + String(currentMoisture) + "%</div>";

  html += "<div class='status-item'><span class='status-label'>Recommended
moisture:</span> " + String(settings.recommendedMoisture) + "%</div>";

  html += "<div class='status-item'><span class='status-label'>Target moisture for
watering:</span> " + String(settings.recommendedMoisture + settings.moistureBuffer) +
"%</div>";

  html += "<div class='status-item'><span class='status-label'>Plant:</span> " +
String(settings.plantName) + "</div>";

  char timeStr[30] = "Never";
  if (settings.lastWateringTime > 0) {
    struct tm *timeinfo;
    timeinfo = localtime(&settings.lastWateringTime);
    strftime(timeStr, sizeof(timeStr), "%d.%m.%Y %H:%M:%S", timeinfo);
  }
  html += "<div class='status-item'><span class='status-label'>Last watering:</span> "
+ String(timeStr) + "</div>";

  html += "<div class='status-item'><span class='status-label'>Check interval:</span>
" + String(settings.checkInterval) + " min</div>";

  html += "</div>";

  html += "<h2>Control</h2>";
  html += "<form id='checkForm' action='/check' method='post'>";
  html += "<button type='button' class='btn' onclick='confirmAction(\"Are you sure you
want to check moisture now?\", \"checkForm\")'>Check moisture now</button>";
  html += "</form>";

  html += "<form id='waterForm' action='/water' method='post'>";
```

```
  html += "<button type='button' class='btn btn-danger' onclick='confirmAction(\"Are
you sure you want to water now?\", \"waterForm\")'>Water now</button>";
  html += "</form>";

  html += "<form id='homeForm' action='/home' method='post'>";
  html += "<button type='button' class='btn' onclick='confirmAction(\"Are you sure you
want to return robot to home position?\", \"homeForm\")'>Return to home
position</button>";
  html += "</form>";

  html += "<h2>Settings</h2>";
  html += "<form id='settingsForm' action='/settings' method='post' onsubmit='return
validateSettings()'>";

  html += "<div class='form-group'>";
  html += "<label for='plantName'>Plant name:</label>";
  html += "<input type='text' id='plantName' name='plantName' value='" +
String(settings.plantName) + "' maxlength='50'>";
  html += "</div>";

  html += "<div class='form-group'>";
  html += "<label for='recommendedMoisture'>Recommended moisture (%):</label>";
  html += "<input type='number' id='recommendedMoisture' name='recommendedMoisture'
min='10' max='90' value='" + String(settings.recommendedMoisture) + "'>";
  html += "</div>";

  html += "<div class='form-group'>";
  html += "<label for='moistureBuffer'>Moisture buffer for watering (%):</label>";
  html += "<input type='number' id='moistureBuffer' name='moistureBuffer' min='0'
max='30' value='" + String(settings.moistureBuffer) + "'>";
  html += "</div>";

  html += "<div class='form-group'>";
  html += "<label for='checkInterval'>Check interval (minutes):</label>";
  html += "<input type='number' id='checkInterval' name='checkInterval' min='5'
max='1440' value='" + String(settings.checkInterval) + "'>";
  html += "</div>";

  html += "<input type='hidden' id='adminPassword' name='adminPassword' value=''>";

  html += "<button type='submit' class='btn'>Save settings</button>";
  html += "</form>";

  html += "<h2>Log</h2>";
  html += "<table class='log-table'>";
  html += "<tr><th>Time</th><th>Moisture</th><th>Event</th></tr>";

  for (int i = 0; i < 10; i++) {
    int idx = (logIndex - i - 1 + 10) % 10;

    if (moistureLog[idx].timestamp > 0) {
      char timeStr[30];
      struct tm *timeinfo;
```

```
      timeinfo = localtime(&moistureLog[idx].timestamp);
      strftime(timeStr, sizeof(timeStr), "%d.%m.%Y %H:%M:%S", timeinfo);

      html += "<tr>";
      html += "<td>" + String(timeStr) + "</td>";
      html += "<td>" + String(moistureLog[idx].moistureLevel) + "%</td>";
      html += "<td>" + String(moistureLog[idx].isWatering ? "Watering" :
"Measurement") + "</td>";
      html += "</tr>";
    }
  }

  html += "</table>";

  html += htmlFooter;
  server.send(200, "text/html", html);
  Serial.println("Main page sent");
}

void handleCheck() {
  Serial.println("Request: moisture check");
  manualCheckRequested = true;
  server.sendHeader("Location", "/");
  server.send(303);
}

void handleWater() {
  Serial.println("Request: watering");
  manualWateringRequested = true;
  server.sendHeader("Location", "/");
  server.send(303);
}

void handleHome() {
  Serial.println("Request: return to home position");
  if (currentState == IDLE || currentState == ERROR) {
    returnHome();
  }
  server.sendHeader("Location", "/");
  server.send(303);
}

void handleSettings() {
  Serial.println("Request: change settings");

  if (server.hasArg("plantName") && server.hasArg("recommendedMoisture") &&
      server.hasArg("moistureBuffer") && server.hasArg("checkInterval")) {

    String plantName = server.arg("plantName");
    int recommendedMoisture = server.arg("recommendedMoisture").toInt();
    int moistureBuffer = server.arg("moistureBuffer").toInt();
    int checkInterval = server.arg("checkInterval").toInt();
    String adminPassword = server.arg("adminPassword");
```

```
    Serial.print("New settings: plant=");
    Serial.print(plantName);
    Serial.print(", recommended moisture=");
    Serial.print(recommendedMoisture);
    Serial.print("%, moisture buffer=");
    Serial.print(moistureBuffer);
    Serial.print("%, check interval=");
    Serial.print(checkInterval);
    Serial.println(" min");

    if (recommendedMoisture < 10 || recommendedMoisture > 90) {
      Serial.println("Error: recommended moisture must be between 10% and 90%");
      server.send(400, "text/plain", "Error: Recommended moisture must be between 10%
and 90%");
      return;
    }

    if (moistureBuffer < 0 || moistureBuffer > 30) {
      Serial.println("Error: moisture buffer must be between 0% and 30%");
      server.send(400, "text/plain", "Error: Moisture buffer must be between 0% and
30%");
      return;
    }

    if (moistureBuffer > 10 && adminPassword != ADMIN_PASSWORD) {
      Serial.println("Error: incorrect admin password");
      server.send(403, "text/plain", "Error: Incorrect admin password");
      return;
    }

    if (checkInterval < 5 || checkInterval > 1440) {
      Serial.println("Error: check interval must be between 5 and 1440 minutes");
      server.send(400, "text/plain", "Error: Check interval must be between 5 and 1440
minutes");
      return;
    }

    strncpy(settings.plantName, plantName.c_str(), 50);
    settings.plantName[50] = '\0';
    settings.recommendedMoisture = recommendedMoisture;
    settings.moistureBuffer = moistureBuffer;
    settings.checkInterval = checkInterval;

    saveSettings();
    Serial.println("Settings successfully saved");
  }

  server.sendHeader("Location", "/");
  server.send(303);
}

void handleNotFound() {
```

```
  Serial.print("Request: page not found - ");
  Serial.println(server.uri());
  server.send(404, "text/plain", "Page not found");
}
```

ДОДАТОК В

Демонстраційний матеріал у вигляді презентації

№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК. 241215.025 ПЗ	Пояснювальна записка	А4, 78 с.
		Додаткові матеріали	
2		Лістинг основного коду	А4, 10 с.
		системи орошення	
3		Лістинг коду веб-інтерфейсу	А4, 8 с.
		та обробників HTTP запитів	
4		Демонстраційний матеріал у	А4, 13 с.
		вигляді презентації	

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК.241215.025 ВД			
Розроб.		Павленко Є.Ю.			Розроблення мобільного робота для автоматизованого поливу рослин у теплицях	Літера	Аркуш	Аркушів
Перевір.		Євсєєв В.В.				Н	1	1
						Кафедра КІТАМ ХНУРЕ		
Н.контр.		Стародубцев М.Г.						
Затв.		Невлюдов І.Ш.						