ДОДАТОК А

Апробація результатів роботи

Міністерство освіти і науки України

NURE

Харківський національний університет радіоелектроніки

ЗБІРНИК

студентських наукових статей

«Автоматизація та приладобудування»

«Automation and Development of Electronic Devices»

ADED-2025

(Випуск 1)

[електронне видання]

KITAP

http://nure.ua/department/kafedra-komp-yuterno-integrovanih-tehnologiy-avtomatizatsiyi-ta-mehatroniki-kitam

iTeD

http://itez.zntu.edu.ua/

http://kafea.kdu.edu.ua

Харків 2025

УДК 621.321

ADVANTAGES AND DISADVANTAGES OF SURFACE ROBOTS IN VARIOUS FIELDS OF APPLICATION

V.A. Kotenko
Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, Nauky Ave. 14
E-mail: volodymyr.kotenko@nure.ua

The article examines the main advantages and disadvantages of surface robots used in scientific, industrial, civilian, and military domains. The study highlights benefits such as operational safety, cost-effectiveness, autonomy, and design features. It also addresses key limitations, including technical challenges, communication issues, legal regulations, and financial barriers. This analysis provides a comprehensive understanding of the potential and limitations of unmanned surface vehicles as an advanced technological solution.

Key words: surface robots, water logistics, advantages and disadvantages, scientific research, monitoring

ПЕРЕВАГИ ТА НЕДОЛІКИ НАДВОДНИХ РОБОТИЗОВАНИХ СИСТЕМ У СУЧАСНИХ СФЕРАХ ЗАСТОСУВАННЯ

В. А. Котенко
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
E-mail: volodymyr.kotenko@nure.ua

У статті розглянуто основні переваги та недоліки використання надводних роботів у різних сферах: науковій, промисловій, цивільній та військовій. Виокремлено переваги, пов'язані з безпекою, економічністю, автономністю та конструктивними особливостями. Також проаналізовано ключові недоліки, включно з технічними складнощами, проблемами зв'язку, правовим регулюванням та фінансовими бар'єрами. Представлений аналіз дозволяє сформувати уявлення про потенціал та обмеження використання надводних роботів.

Ключові слова: надводні роботи, водна логістика, переваги та недоліки, наукові дослідження, моніторинг.

RELEVANCE OF THE WORK. In era of modern technological advancements and requirements faced by scientific, military, and industrial sectors, surface robots are gaining increasing popularity as a versatile tool for performing various tasks. Their ability to operate in challenging conditions, ensure autonomy, and reduce human factors opens new possibilities for monitoring water bodies, conducting research, and ensuring security. Surface robots are becoming important participants not only in rescue and military operations but also in fields such as ecology, climate change research, and natural resource management. The use of these technologies significantly increases task efficiency, reduces costs and risks, and ensures accuracy in data collection and processing.

INTRODUCTION. Surface robots are gaining increasing importance in various fields due to their ability to perform complex tasks and operate in conditions where human presence is undesirable or dangerous. An important part of their development and implementation is understanding both their advantages and disadvantages. Considering these aspects is necessary for creating effective and safe systems, as well as minimizing risks and improving the use of surface robots in various conditions. This article will examine the main advantages and disadvantages of such technologies, allowing for an assessment of their potential and identification of key factors for successful implementation.

MATERIALS AND RESEARCH RESULTS. One of the main advantages of surface robots is their ability to perform dangerous missions without involving human crews. In particular, when clearing water areas of mines, protecting coastal zones, or operating in adverse weather conditions, these devices help avoid risks to human lives. Additionally, such devices can be used to target water-based objectives, which is of particular importance in the context of modern military conflicts [1].

The use of surface robots significantly reduces economic costs. The absence of crew members eliminates the need for life support systems, reduces fuel costs, technical maintenance and repair expenses for equipment and the device itself, as well as employee salaries [2].

Surface robots enable the implementation of long-term tasks since they can remain in aquatic environments much longer than vessels with human crews. The absence of human factors—such as the need for sleep, food, etc.—provides the possibility of continuous operation for days, weeks, and even months or years. Furthermore, such devices can function in harsher weather conditions compared to conventional vessels since there is no need to consider crew comfort [3].

In the absence of a crew and life support systems, surface robots can have a more aerodynamic design, making them faster, more maneuverable, and smaller in size. These characteristics make controlling such devices easier, allow for installation of more equipment necessary for achieving set objectives, and enable access to hard-to-reach places [4].

Surface robots play a key role in modernizing the field of maritime transportation and logistics, ensuring high levels of productivity and cost reduction. Thanks to autonomous operation without the need for a crew, these devices significantly reduce maintenance costs and eliminate risks to personnel health and life. Equipped with intelligent navigation and communication systems, they can effectively navigate maritime routes, avoid obstacles, and build optimal movement trajectories, positively impacting fuel economy and reducing harmful emissions into the environment. Additionally, surface robots can perform functions of remote monitoring of vessel technical conditions, cargo inspection, and surveillance of port facilities, contributing to increased safety of maritime operations [5].

Modern surface robots demonstrate significant environmental advantages compared to traditional vessels. Due to their compact size and lightweight construction, they consume less fuel, leading to reduced greenhouse gas emissions. Many models are equipped with alternative energy sources, such as wind and solar installations, allowing them to operate for extended periods without refueling, reducing dependence on fossil fuels. Furthermore, the absence of the need to transport fuel or other hazardous materials reduces the risk of environmental disasters, such as oil spills. These characteristics make surface robots more environmentally safe and contribute to the sustainable development of maritime operations [5].

However, like any technical means, surface robots have certain disadvantages. The first is the constant exposure to moisture, salt and fresh water, and temperature fluctuations, which creates specific requirements for protecting electronic and mechanical elements. Water corrosion can quickly damage unprotected elements, significantly reducing their service life and that of the entire system. The need to create waterproof protection significantly increases production costs and adds complexity to the device's construction. At the same time, the requirement for complete sealing of all components creates obstacles for quick repairs and scheduled maintenance, causing even minor technical problems to lead to interruptions in the robot's operation [5].

Another disadvantage of surface robots is the vulnerability of communication and navigation systems during cyber attacks or electronic countermeasures. During mission execution, navigation problems may arise due to interception of control over the device or replacement of navigation data, which can not only disrupt the task but also potentially make the device a dangerous tool against those who use it. Therefore, creating an effective security system for communication channels and software becomes critically necessary, although technically extremely complex [6].

The legislative framework regarding surface robots is still forming, and there is a need for standardized rules for their safe operation. Regulatory aspects regarding licensing, insurance mechanisms, legal responsibility, and safety certification require comprehensive regulatory settlement for facilitating full-scale integration into maritime operational systems. The lack of approved legal norms may become a restraining factor for the use of surface robots [7].

Despite the fact that surface robots provide savings by reducing operational costs, initial investments in their acquisition, repair, and modernization can be significant. The cost of modern technologies, namely sensors, communication systems, and artificial intelligence integration, additionally increases the overall cost of the robot. Price affordability may be a limiting factor, especially for clients or organizations with limited budgets [7].

Another significant limitation of surface robots is their limited payload capacity. Due to compact sizes and design features, these devices cannot transport heavy equipment or large volumes of cargo, limiting their application in certain areas, such as transporting significant amounts of materials. This limitation needs to be considered when planning missions and determining tasks that can be effectively performed using surface robots [7].

The complexity of integration with existing infrastructure and management systems is a serious challenge for the widespread implementation of surface robots in navigation. Most existing ports, navigation hubs, and logistics facilities are designed to service traditional vessels with human crews. They are not adapted to autonomous devices that require special conditions for mooring, recharging, cargo transfer, or data reading. Additionally, existing management and monitoring systems often do not support communication protocols used by surface robots. This leads to incompatibility at the level of software and communication channels, complicating centralized management of unmanned device fleets. Overcoming these problems requires modernization of infrastructure, including installation of appropriate terminals, automated docks, charging stations, as well as integration of unified information systems. All this requires significant financial and technical resources, which may become a restraining factor for many countries and companies, especially in conditions of limited budget or technical backwardness of the region [8].

CONCLUSIONS. Surface robots play an increasingly significant role in modern scientific, industrial, and security fields due to their autonomy, environmental friendliness, and ability to effectively operate in difficult conditions without human risk. They demonstrate a wide spectrum of advantages—from reducing costs and minimizing human factors to increasing accuracy and duration of task execution. Along with this, it is important to consider technical and regulatory limitations associated with their operation, among which the influence of the aquatic environment, cyber threats, high technology costs, and limited payload capacity should be especially highlighted. Further development of this field requires improvement of designs, enhancement of security system reliability, and formation of a unified legal framework that will allow for full integration of surface robots into various spheres of human activity.

REFERENCES

1. Ghazali M. H. M. Unmanned surface vehicles: from a hull design perspective [Electronic resource] / Mohamad Hazwan Mohd Ghazali, Mohd Hafiz Abdul Satar, Wan Rahiman. – 2024. – Vol. 312. – P. 118977. – Mode of access: https://doi.org/10.1016/j.oceaneng.2024.118977

2. Kolawole A. Review of the role of unmanned surface vehicles (usvs) in enhancing offshore oil and gas exploration: opportunities and challenges in nigerian waters" [Electronic resource]. – 2024. – Vol. IX, no. X. – P. 171–186. – Mode of access: https://doi.org/10.51584/ijrias.2024.910017

3. The milestone underscores the reliability of the vehicle, and confirms Saildrone's position as the only proven long-range, long-endurance uncrewed platform. 2023 [Electronic resource]. – Mode of

access: https://www.saildrone.com/media-room/press-releases/saildrone-fleet-surpasses-1000000-nautical-miles-32-000-days-at-sea (date of access: 17.04.2025). – Title from screen.

4. Unmanned surface vessels: the catamaran advantage | defense.info [Electronic resource]. – Mode of access: https://defense.info/maritime-dynamics/2023/02/unmanned-surface-vessels-the-catamaran-advantage/ (date of access: 17.04.2025). – Title from screen.

5. Selecting a conformal coating for marine electronics [Electronic resource]. – Mode of access: https://www.unmannedsystemstechnology.com/feature/selecting-a-conformal-coating-for-marine-electronics/ (date of access: 17.04.2025). – Title from screen.

6. Are uncrewed surface vehicles the future of naval combat? [Electronic resource]. – Mode of access: https://www.naval-technology.com/analyst-comment/uncrewed-surface-vehicles-future-naval-combat/ (date of access: 17.04.2025). – Title from screen.

7. Uncrewed surface vehicles: revolutionizing maritime operations - terminautical [Electronic resource]. – Mode of access: https://terminautical.com/uncrewed-surface-vehicles-revolutionizing-maritime-operations/ (date of access: 17.04.2025). – Title from screen.

8. Elyamany N. A. Posthuman emotion artificial intelligence in postcyberpunk cityscape: a multimodal reading of blade runner 2049 [Electronic resource]. – 2021. – Vol. 1, no. 1. – P. 04. – Mode of access: https://doi.org/10.21622/ilcc.2021.01.1.004

9. Vladyslav, Y., & Bronnikov, A. (2020, October). ANALYSIS OF THE CMMI MODEL APPLICATION FOR SOLVING THE TASKS OF CPPS CONTROL PROCESSES AUTOMATION DEVELOPMENT. In The 4 th International scientific and practical conference "Actual trends of modern scientific research"(October 11-13, 2020) MDPC Publishing, Munich, Germany. 2020. 386 p. (p. 128).

10. Yevsieiev, V. V., & Bronnikov, A. I. (2020). Development of databases interconnection "essences" information model for cyber-physical production systems additive cyber design creation automation. Збірник Наукових Праць НУК, №3. С.56-62. DOI https://doi.org/10.15589/znp2020.3(481).7

11. Yevsieiev V., Bronnikov A. Information systems development methodologies application analysis for cyber-physical production systems development. III International scientific-practical conference "Theory, science and practice" (Japan, Tokyo, 5–8 October 2020). P. 398–401. DOI: 10.46299/ISG.2020.II.III

12. Yevsieiev V., Bronnikov A. Analysis of the cyber-physical production systems implementation impact to achieve the goals of lean production. The IIth International scientific and practical conference «Development of scientific and practical approaches in the era of globalization» (USA, Boston, 28–30 September. 2020). P.221–226. DOI:10.46299/ISG.2020.II.II.

13. Attar, H., & et al.. (2022). Zoomorphic Mobile Robot Development for Vertical Movement Based on the Geometrical Family Caterpillar. Computational Intelligence and Neuroscience, 2022, Article ID 3046116, https://doi.org/10.1155/2022/3046116.

14. Abu-Jassar AT, Attar H, Amer A, et al. Remote Monitoring System of Patient Status in Social IoT Environments Using Amazon Web Services (AWS) Technologies and Smart Health Care. International Journal of Crowd Science, 2024, https://doi.org/10.26599/IJCS.2023.9100019

15. Abu-Jassar, A. T., Attar, H., Yevsieiev, V., Amer, A., Demska, N., Luhach, A. K., & Lyashenko, V. (2022). Electronic user authentication key for access to HMI/SCADA via unsecured internet networks. *Computational intelligence and neuroscience*, *2022*(1), 5866922.

***Науковий керівник**: Бронніков Артем Ігорович, к.т.н., доц., доцент кафедри КІТАР Харківського національного університету радіоелектроніки*

ДОДАТОК Б

Програмний код для ESP32-CAM

```
#include "esp_camera.h"
#include <Arduino.h>
#include <WiFi.h>
#include <ESPAsyncWebServer.h>
#include <AsyncTCP.h>

#define FLASH_LED_PIN 4

// Змінні для зберігання останніх значень відстаней (отримуються від Dev Kit)
float frontDistance = 0;
float leftDistance = 0;
float rightDistance = 0;
unsigned long lastSensorUpdate = 0;

// Змінні для системи індикації перешкод
bool frontWarningShown = false;
bool leftWarningShown = false;
bool rightWarningShown = false;
const float SAFE_DISTANCE = 200.0;    // см - зелений діапазон 400-200
const float WARNING_DISTANCE = 100.0; // см - жовтий діапазон 200-100
// Червоний діапазон 100-0

// Зміщення датчиків від країв робота (см)
const float FRONT_SENSOR_OFFSET = 27.0; // Передній датчик зміщений на 27 см
const float LEFT_SENSOR_OFFSET = 3.0;   // Лівий датчик зміщений на 3 см
const float RIGHT_SENSOR_OFFSET = 3.0;  // Правий датчик зміщений на 3 см

// Camera GPIO Configuration
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

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

AsyncWebServer server(80);

const char* htmlHomePage PROGMEM = R"HTMLHOMEPAGE(
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-
scale=1, user-scalable=no">
    <meta charset="UTF-8">
    <style>
      * {
        box-sizing: border-box;
        margin: 0;
        padding: 0;
        user-select: none;
        -webkit-user-select: none;
        -moz-user-select: none;
        -ms-user-select: none;
      }
      body {
        font-family: 'Arial', sans-serif;
        background-color: #f0f4f8;
        display: flex;
        justify-content: center;
        align-items: center;
        min-height: 100vh;
        padding: 20px;
      }
      .container {
        background-color: white;
        border-radius: 15px;
        box-shadow: 0 10px 25(
        padding: 20px;
        max-width: 450px;
        width: 100%;
      }
      .page-title {
        font-size: 24px;
        font-weight: bold;
        color: #2c3e50;
        text-align: center;
        margin-bottom: 20px;
        padding-bottom: 10px;
        border-bottom: 2px solid #4a90e2;
      }
      .camera-feed {
        width: 100%;
        height: 250px;
        background-color: #e9ecef;
        border-radius: 10px;
        margin-bottom: 20px;
        overflow: hidden;
      }
      .camera-feed img {
        width: 100%;
        height: 100%;
        object-fit: cover;
      }
      .control-grid {
        display: grid;
        grid-template-columns: repeat(3, 1fr);
```

```
  grid-template-areas:
    ". up    ."
    "left stop right"
    ". down  .";
  gap: 12px;
  margin-bottom: 20px;
  align-items: center;
  justify-content: center;
}
.control-grid button:nth-child(1) { grid-area: up; }
.control-grid button:nth-child(2) { grid-area: left; }
.control-grid button:nth-child(3) { grid-area: stop; }
.control-grid button:nth-child(4) { grid-area: right; }
.control-grid button:nth-child(5) { grid-area: down; }
.control-btn {
  background: linear-gradient(145deg, #5ba3f5, #4a90e2);
  color: white;
  border: none;
  border-radius: 12px;
  padding: 18px;
  font-size: 20px;
  font-weight: bold;
  cursor: pointer;
  transition: all 0.2s ease;
  outline: none;
  box-shadow: 0 4px 8px rgba(74, 144, 226, 0.3);
  position: relative;
  overflow: hidden;
}
.control-btn:active {
  background: linear-gradient(145deg, #357abd, #2e6aa1);
  transform: translateY(2px);
  box-shadow: 0 2px 4px rgba(74, 144, 226, 0.4);
}
.control-btn:hover {
  transform: translateY(-1px);
  box-shadow: 0 6px 12px rgba(74, 144, 226, 0.4);
}
.slider-container {
  display: flex;
  flex-direction: column;
  gap: 8px;
  margin-bottom: 15px;
  position: relative;
}
.slider-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 5px;
}
.slider-label {
  font-weight: bold;
  color: #2c3e50;
  display: flex;
  align-items: center;
  gap: 8px;
}
.slider-value {
```

```
  font-weight: bold;
  color: #4a90e2;
  font-size: 16px;
  min-width: 45px;
  text-align: right;
}
.slider-wrapper {
  position: relative;
  display: flex;
  align-items: center;
  gap: 10px;
}
.direction-arrow {
  font-size: 18px;
  color: #4a90e2;
  font-weight: bold;
}
.slider {
  flex: 1;
  -webkit-appearance: none;
  appearance: none;
  height: 12px;
  background: #e0e0e0;
  border-radius: 6px;
  outline: none;
  opacity: 0.8;
  transition: opacity 0.2s;
  position: relative;
}
.slider:hover {
  opacity: 1;
}
.slider::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  width: 24px;
  height: 24px;
  background: linear-gradient(145deg, #5ba3f5, #4a90e2);
  border-radius: 50%;
  cursor: pointer;
  box-shadow: 0 2px 6px rgba(74, 144, 226, 0.4);
  transition: all 0.2s ease;
}
.slider::-webkit-slider-thumb:hover {
  transform: scale(1.1);
  box-shadow: 0 3px 8px rgba(74, 144, 226, 0.6);
}
.slider::-moz-range-thumb {
  width: 24px;
  height: 24px;
  background: linear-gradient(145deg, #5ba3f5, #4a90e2);
  border-radius: 50%;
  cursor: pointer;
  border: none;
  box-shadow: 0 2px 6px rgba(74, 144, 226, 0.4);
}
.center-mark {
  position: absolute;
  top: 50%;
```

```
  left: 50%;
  transform: translate(-50%, -50%);
  width: 3px;
  height: 16px;
  background-color: #e74c3c;
  border-radius: 2px;
  pointer-events: none;
}
.sensors-panel {
  background-color: #e8f5e9;
  border-radius: 10px;
  padding: 15px;
  margin-bottom: 20px;
}
.sensors-title {
  font-weight: bold;
  color: #2c3e50;
  text-align: center;
  margin-bottom: 10px;
}
.sensors-grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 10px;
  text-align: center;
}
.sensor-box {
  padding: 8px;
  border-radius: 8px;
  background-color: white;
  box-shadow: 0 2px 5px rgba(0,0,0,0.1);
  display: flex;
  flex-direction: column;
  align-items: center;
  position: relative;
}
.sensor-label {
  font-size: 12px;
  color: #555;
  margin-bottom: 5px;
}
.sensor-value {
  font-size: 18px;
  font-weight: bold;
  color: #2e7d32;
  margin-bottom: 8px;
}
.sensor-icon {
  font-size: 20px;
  margin-bottom: 5px;
  color: #4a90e2;
}

/* Статус індикатори - кружечки */
.status-indicator {
  position: absolute;
  top: 5px;
  right: 5px;
  width: 12px;
```

```
  height: 12px;
  border-radius: 50%;
  border: 2px solid white;
  box-shadow: 0 1px 3px rgba(0,0,0,0.3);
}

.status-indicator.safe {
  background-color: #4CAF50;
}

.status-indicator.warning {
  background-color: #FF9800;
  animation: statusBlink 1.5s ease-in-out infinite;
}

.status-indicator.danger {
  background-color: #F44336;
  animation: statusBlink 0.8s ease-in-out infinite;
}

@keyframes statusBlink {
  0%, 100% { opacity: 1; }
  50% { opacity: 0.3; }
}

/* Полоски внизу датчиків */
.safety-indicator {
  width: 100%;
  height: 6px;
  border-radius: 3px;
  background-color: #ddd;
  overflow: hidden;
  position: relative;
  margin-top: 5px;
}

.safety-bar {
  height: 100%;
  border-radius: 3px;
  transition: all 0.3s ease;
  position: relative;
}

.safety-bar.safe {
  background: linear-gradient(90deg, #4CAF50, #81C784);
  width: 100%;
}

.safety-bar.warning {
  background: linear-gradient(90deg, #FF9800, #FFB74D);
  width: 100%;
  animation: warningPulse 1.5s ease-in-out infinite;
}

.safety-bar.danger {
  background: linear-gradient(90deg, #F44336, #EF5350);
  width: 100%;
  animation: dangerPulse 1s ease-in-out infinite;
}
```

```
      @keyframes warningPulse {
        0%, 100% { opacity: 1; }
        50% { opacity: 0.6; }
      }

      @keyframes dangerPulse {
        0%, 100% { opacity: 1; transform: scaleY(1); }
        50% { opacity: 0.8; transform: scaleY(1.2); }
      }
    </style>
  </head>
  <body>
    <div class="container">
      <h1 class="page-title">🚤 Надводний Робот</h1>
      <div class="camera-feed">
        <img id="cameraImage" src="/camera_feed" alt="Camera Feed">
      </div>

      <div class="sensors-panel">
        <div class="sensors-title">Відстань до перешкоди</div>
        <div class="sensors-grid">
          <div class="sensor-box">
            <div class="status-indicator safe" id="leftStatus"></div>
            <div class="sensor-icon">←</div>
            <div class="sensor-label">Зліва</div>
            <div class="sensor-value" id="leftDistance">-- см</div>
            <div class="safety-indicator">
              <div class="safety-bar safe" id="leftSafetyBar"></div>
            </div>
          </div>
          <div class="sensor-box">
            <div class="status-indicator safe" id="frontStatus"></div>
            <div class="sensor-icon">↑</div>
            <div class="sensor-label">Спереду</div>
            <div class="sensor-value" id="frontDistance">-- см</div>
            <div class="safety-indicator">
              <div class="safety-bar safe" id="frontSafetyBar"></div>
            </div>
          </div>
          <div class="sensor-box">
            <div class="status-indicator safe" id="rightStatus"></div>
            <div class="sensor-icon">→</div>
            <div class="sensor-label">Справа</div>
            <div class="sensor-value" id="rightDistance">-- см</div>
            <div class="safety-indicator">
              <div class="safety-bar safe" id="rightSafetyBar"></div>
            </div>
          </div>
        </div>
      </div>

      <div class="control-grid">
        <button class="control-btn" onmousedown="sendCommand('MoveRobot', 1)"
onmouseup="sendCommand('MoveRobot', 0)" ontouchstart="sendCommand('MoveRobot',
1)" ontouchend="sendCommand('MoveRobot', 0)">▲</button>
```

```
        <button class="control-btn" onmousedown="sendCommand('MoveRobot', 3)"
onmouseup="sendCommand('MoveRobot', 0)" ontouchstart="sendCommand('MoveRobot',
3)" ontouchend="sendCommand('MoveRobot', 0)">◄</button>
        <button class="control-btn" onmousedown="sendCommand('MoveRobot', 0)"
ontouchstart="sendCommand('MoveRobot', 0)">■</button>
        <button class="control-btn" onmousedown="sendCommand('MoveRobot', 4)"
onmouseup="sendCommand('MoveRobot', 0)" ontouchstart="sendCommand('MoveRobot',
4)" ontouchend="sendCommand('MoveRobot', 0)">►</button>
        <button class="control-btn" onmousedown="sendCommand('MoveRobot', 2)"
onmouseup="sendCommand('MoveRobot', 0)" ontouchstart="sendCommand('MoveRobot',
2)" ontouchend="sendCommand('MoveRobot', 0)">▼</button>
      </div>

      <div class="slider-container">
        <div class="slider-header">
          <div class="slider-label">📹 Поворот камери</div>
          <div class="slider-value" id="panValue">90°</div>
        </div>
        <div class="slider-wrapper">
          <input type="range" min="0" max="180" value="90" class="slider"
id="Pan" oninput="updatePanValue(this.value); sendCommand('Pan', this.value)">
          <div class="center-mark"></div>
        </div>
      </div>

      <div class="slider-container">
        <div class="slider-header">
          <div class="slider-label">🔍 Опускання ехолота</div>
          <div class="slider-value" id="tiltValue">0°</div>
        </div>
        <div class="slider-wrapper">
          <div class="direction-arrow">▲</div>
          <input type="range" min="0" max="180" value="0" class="slider"
id="Tilt" oninput="updateTiltValue(this.value); sendCommand('Tilt', this.value)">
          <div class="direction-arrow">▼</div>
        </div>
      </div>

      <div class="slider-container">
        <div class="slider-header">
          <div class="slider-label">⚡ Швидкість</div>
          <div class="slider-value" id="speedValue">150</div>
        </div>
        <div class="slider-wrapper">
          <input type="range" min="0" max="255" value="150" class="slider"
id="Speed" oninput="updateSpeedValue(this.value); sendCommand('Speed',
this.value)">
        </div>
      </div>
    </div>
     <script>
      // Функція для відправки HTTP-запитів замість WebSocket
      function sendCommand(command, value) {
         if (command === 'Pan')
        var xhr = new XMLHttpRequest();
        xhr.open("GET", "/control?command=" + command + "&value=" + value, true);
        xhr.send();
      }
```

```
      // Функції для оновлення відображення значень слайдерів
      function updatePanValue(value) {
        document.getElementById("panValue").innerHTML = value + "°";
      }

      function updateTiltValue(value) {
        document.getElementById("tiltValue").innerHTML = value + "°";
      }

      function updateSpeedValue(value) {
        document.getElementById("speedValue").innerHTML = value;
      }

      // Функція для визначення рівня безпеки на основі відстані
      // Нові діапазони: 400-200см зелений, 200-100см жовтий, 100-0см червоний
      function getSafetyLevel(distance) {
        if (distance <= 0) return 'unknown';

        if (distance >= 200) return 'safe';      // Зелений - 400-200см
        if (distance >= 100) return 'warning';   // Жовтий - 200-100см
        return 'danger';                         // Червоний - 100-0см
      }

      // Функція для оновлення індикаторів безпеки
      function updateSafetyIndicator(prefix, distance) {
        var safetyLevel = getSafetyLevel(distance);
        var safetyBar = document.getElementById(prefix + "SafetyBar");
        var statusIndicator = document.getElementById(prefix + "Status");

        // Очищаємо всі класи
        safetyBar.className = "safety-bar";
        statusIndicator.className = "status-indicator";

        // Додаємо відповідний клас
        if (safetyLevel !== 'unknown') {
          safetyBar.classList.add(safetyLevel);
          statusIndicator.classList.add(safetyLevel);
        } else {
          safetyBar.classList.add('safe');
          statusIndicator.classList.add('safe');
        }
      }

      // Функція для періодичного оновлення даних датчиків
      function updateSensorData() {
        var xhr = new XMLHttpRequest();
        xhr.onreadystatechange = function() {
          if (this.readyState == 4 && this.status == 200) {
            var data = JSON.parse(this.responseText);

            // Відображаємо цілі числа
            document.getElementById("frontDistance").innerHTML =
Math.round(data.frontDistance) + " см";
            document.getElementById("leftDistance").innerHTML =
Math.round(data.leftDistance) + " см";
            document.getElementById("rightDistance").innerHTML =
Math.round(data.rightDistance) + " см";

            // Оновлюємо індикатори з новими діапазонами
```

```
            updateSafetyIndicator("front", data.frontDistance);
            updateSafetyIndicator("left", data.leftDistance);
            updateSafetyIndicator("right", data.rightDistance);
          }
        };
        xhr.open("GET", "/sensor_data", true);
        xhr.send();
      }

      // Функція для оновлення зображення з камери
      function refreshCameraImage() {
        var img = document.getElementById("cameraImage");
        img.src = "/camera_feed?t=" + new Date().getTime(); // Додавання часової
мітки для запобігання кешування
      }

      // Ініціалізація при завантаженні сторінки
      window.onload = function() {
        // Відправляємо початкові значення слайдерів
        sendCommand("Speed", document.getElementById("Speed").value);
        sendCommand("Pan", document.getElementById("Pan").value);
        sendCommand("Tilt", document.getElementById("Tilt").value);

        // Ініціалізуємо відображення значень
        updatePanValue(document.getElementById("Pan").value);
        updateTiltValue(document.getElementById("Tilt").value);
        updateSpeedValue(document.getElementById("Speed").value);

        // Періодичне оновлення даних сенсорів (кожні 500 мс)
        setInterval(updateSensorData, 500);

        // Періодичне оновлення зображення з камери (кожні 200 мс)
        setInterval(refreshCameraImage, 200);
      };
    </script>
  </body>
</html>
)HTMLHOMEPAGE";

// Функція для парсингу даних сенсорів, отриманих від Dev Kit
void parseSensorData(String data) {
  // Формат: "SENSORS,front,left,right" наприклад "SENSORS,25.3,30.1,15.7"
  if (data.startsWith("SENSORS,")) {
    data = data.substring(8); // Видаляємо "SENSORS,"

    int firstComma = data.indexOf(',');
    int secondComma = data.indexOf(',', firstComma + 1);

    if (firstComma > 0 && secondComma > 0) {
      // Парсимо відстані
      float rawFrontDistance = data.substring(0, firstComma).toFloat();
      float rawLeftDistance = data.substring(firstComma + 1,
secondComma).toFloat();
      float rawRightDistance = data.substring(secondComma + 1).toFloat();

      // Віднімаємо зміщення і забезпечуємо, щоб відстань не була від'ємною
      frontDistance = std::max(0.0f, rawFrontDistance - FRONT_SENSOR_OFFSET);
      leftDistance = std::max(0.0f, rawLeftDistance - LEFT_SENSOR_OFFSET);
      rightDistance = std::max(0.0f, rawRightDistance - RIGHT_SENSOR_OFFSET);
```

```
      lastSensorUpdate = millis();

      // Перевірка критичних відстаней
      checkAndPrintCompactWarnings();

      Serial.printf("Sensor data (corrected) - F:%.1f L:%.1f R:%.1f\n",
                    frontDistance, leftDistance, rightDistance);
    }
  }
}

// Функція для перевірки критичних відстаней
void checkAndPrintCompactWarnings() {
  String status = "STATUS: ";
  bool hasAlert = false;

  // Перевірка спереду (400-200 зелений, 200-100 жовтий, 100-0 червоний)
  if (frontDistance > 0) {
    if (frontDistance < 100) {
      status += "F:CRIT ";
      hasAlert = true;
    } else if (frontDistance < 200) {
      status += "F:WARN ";
      hasAlert = true;
    }
  }

  // Перевірка зліва (ті ж діапазони)
  if (leftDistance > 0) {
    if (leftDistance < 100) {
      status += "L:CRIT ";
      hasAlert = true;
    } else if (leftDistance < 200) {
      status += "L:WARN ";
      hasAlert = true;
    }
  }

  // Перевірка справа (ті ж діапазони)
  if (rightDistance > 0) {
    if (rightDistance < 100) {
      status += "R:CRIT ";
      hasAlert = true;
    } else if (rightDistance < 200) {
      status += "R:WARN ";
      hasAlert = true;
    }
  }

  // Виводимо статус тільки при зміні або наявності попереджень
  if (hasAlert) {
    Serial.println(status);
  }
}

void sendRobotCommands(String inputCommand) {
  Serial.println("Sending to Dev Kit: " + inputCommand);
  Serial1.println(inputCommand);  // Відправляємо команди на Serial1 (Dev Kit)
```

```
}

void handleRoot(AsyncWebServerRequest *request) {
  request->send_P(200, "text/html", htmlHomePage);
}

void handleControl(AsyncWebServerRequest *request) {
  String command = request->getParam("command")->value();
  String value = request->getParam("value")->value();

  String fullCommand = command + "," + value;
  sendRobotCommands(fullCommand);

  // Відправляємо відповідь, що команда отримана
  request->send(200, "text/plain", "OK");
}

void handleSensorData(AsyncWebServerRequest *request) {
  String jsonResponse = "{\"frontDistance\":" + String(frontDistance, 1) +
                        ",\"leftDistance\":" + String(leftDistance, 1) +
                        ",\"rightDistance\":" + String(rightDistance, 1) + "}";
  request->send(200, "application/json", jsonResponse);
}

// Обробник для віддачі зображення з камери
void handleCameraCapture(AsyncWebServerRequest *request) {
  camera_fb_t * fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Camera capture failed");
    request->send(500, "text/plain", "Camera capture failed");
    return;
  }

  AsyncResponseStream *response = request->beginResponseStream("image/jpeg");
  response->addHeader("Content-Disposition", "inline; filename=capture.jpg");
  response->write((const char *)fb->buf, fb->len);
  request->send(response);

  esp_camera_fb_return(fb);
}

void handleNotFound(AsyncWebServerRequest *request) {
  request->send(404, "text/plain", "File Not Found");
}

void setupCamera() {
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
```

```
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 2;

  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  Serial.println("Camera initialized successfully");
  sensor_t * s = esp_camera_sensor_get();
  if (s) {
    // Налаштування параметрів датчика камери
    s->set_brightness(s, 0);     // -2 to 2
    s->set_contrast(s, 0);       // -2 to 2
    s->set_saturation(s, 0);     // -2 to 2
    s->set_special_effect(s, 0); // 0 to 6
    s->set_whitebal(s, 1);       // 0 = disable, 1 = enable
    s->set_awb_gain(s, 1);       // 0 = disable, 1 = enable
    s->set_wb_mode(s, 0);        // 0 to 4 - if awb_gain enabled
    s->set_exposure_ctrl(s, 1);  // 0 = disable, 1 = enable
    s->set_aec2(s, 0);           // 0 = disable, 1 = enable
    s->set_gain_ctrl(s, 1);      // 0 = disable, 1 = enable
    s->set_agc_gain(s, 0);       // 0 to 30
    s->set_gainceiling(s, (gainceiling_t)0);  // 0 to 6
    s->set_bpc(s, 0);            // 0 = disable, 1 = enable
    s->set_wpc(s, 1);            // 0 = disable, 1 = enable
    s->set_raw_gma(s, 1);        // 0 = disable, 1 = enable
    s->set_lenc(s, 1);           // 0 = disable, 1 = enable
    s->set_hmirror(s, 0);        // 0 = disable, 1 = enable
    s->set_vflip(s, 0);          // 0 = disable, 1 = enable
    s->set_dcw(s, 1);            // 0 = disable, 1 = enable
  }
}

void setup(void) {
  Serial.begin(115200);
  pinMode(FLASH_LED_PIN, OUTPUT);
  digitalWrite(FLASH_LED_PIN, LOW);

  // Налаштування Serial для передачі команд на ESP32 DevKit та отримання даних
сенсорів
  // RX=3 (U0RXD), TX=1 (U0TXD) підключаються до GPIO17, GPIO16 Dev Kit
відповідно
  Serial1.begin(115200, SERIAL_8N1, 3, 1);

  // Налаштування WiFi точки доступу
  WiFi.mode(WIFI_AP);
  WiFi.softAP(ssid, password);
  IPAddress IP = WiFi.softAPIP();
```

```
  Serial.print("AP IP address: ");
  Serial.println(IP);

  // Налаштування обробників HTTP-запитів
  server.on("/", HTTP_GET, handleRoot);
  server.on("/control", HTTP_GET, handleControl);
  server.on("/sensor_data", HTTP_GET, handleSensorData);
  server.on("/camera_feed", HTTP_GET, handleCameraCapture);
  server.onNotFound(handleNotFound);

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

  setupCamera();

  Serial.println("ESP32-CAM готовий до роботи!");
}

void loop() {
  // Читання даних від Dev Kit (включно з даними сенсорів)
  if (Serial1.available()) {
    String receivedData = Serial1.readStringUntil('\n');
    receivedData.trim();

    // Якщо отримані дані містять інформацію про сенсори
    if (receivedData.startsWith("SENSORS,")) {
      parseSensorData(receivedData);
    } else {
      Serial.println("Received from Dev Kit: " + receivedData);
    }
  }

  // Передача команд з Serial на Serial1 (для налагодження)
  if (Serial.available()) {
    String command = Serial.readStringUntil('\n');
    Serial1.println(command);
  }
}
```

ДОДАТОК В

Програмний код для ESP32 DevKit V1

```
#include <vector>
#include <iostream>
#include <sstream>
#include <ESP32Servo.h>

// Піни для ультразвукових датчиків
#define TRIGGER_PIN_FRONT 25   // Пін для TRIG переднього датчика
#define ECHO_PIN_FRONT 26      // Пін для ECHO переднього датчика
#define TRIGGER_PIN_LEFT 32    // Пін для TRIG лівого датчика
#define ECHO_PIN_LEFT 33       // Пін для ECHO лівого датчика
#define TRIGGER_PIN_RIGHT 2   // Пін для TRIG правого датчика
#define ECHO_PIN_RIGHT 4      // Пін для ECHO правого датчика

#define PAN_SERVO_PIN 12           // Серводвигун для повороту камери
#define TILT_SERVO_PIN 13          // Серводвигун для опускання ехолота
#define STEERING_SERVO1_PIN 14     // Перший сервопривід керування (ліво)
#define STEERING_SERVO2_PIN 27     // Другий сервопривід керування (право)

// Визначення напрямків руху
#define UP 1
#define DOWN 2
#define LEFT 3
#define RIGHT 4
#define STOP 0

// Налаштування моторів
#define RIGHT_MOTOR 0
#define LEFT_MOTOR 1
#define FORWARD 1
#define BACKWARD -1

const int MAX_DISTANCE = 400; // Максимальна відстань в сантиметрах

// Змінні для зберігання останніх значень відстаней
float frontDistance = 0;
float leftDistance = 0;
float rightDistance = 0;
unsigned long lastDistanceMeasurement = 0;
const int MEASUREMENT_INTERVAL = 500; // Інтервал вимірювань

// Структура для pins моторів
struct MOTOR_PINS {
  int pinIN1;
  int pinIN2;
  int pinEn;
  int pwmSpeedChannel;
};

// Pins для моторів
std::vector<MOTOR_PINS> motorPins = {
```

```
  {5, 18, 22, 4},   // RIGHT_MOTOR
  {19, 21, 23, 5},  // LEFT_MOTOR
};

const int PWMFreq = 1000; // 1 KHz
const int PWMResolution = 8;
int currentSpeed = 150; // Стандартна швидкість

Servo panServo;
Servo tiltServo;
Servo steeringServo1;
Servo steeringServo2;

// Функція вимірювання відстані
float measureDistance(int trigPin, int echoPin) {
  // Очищаємо тригер
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);

  // Відправляємо імпульс на TRIG (10µs high pulse)
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(15);
  digitalWrite(trigPin, LOW);

  // Вимірюємо тривалість високого сигналу на ECHO з таймаутом
  unsigned long timeout = (MAX_DISTANCE * 58) + 1000;
  long duration = pulseIn(echoPin, HIGH, timeout);

  // Розрахунок відстані в сантиметрах
  float distance_cm;
  if (duration > 0) {
    distance_cm = duration / 58.0;
  } else {
    distance_cm = MAX_DISTANCE; // Якщо таймаут - повертаємо максимальну відстань
  }

  // Обмежуємо максимальну відстань
  if (distance_cm > MAX_DISTANCE || distance_cm < 2) {
    distance_cm = MAX_DISTANCE;
  }

  return distance_cm;
}

// Функція для вимірювання всіх трьох датчиків
void measureAllDistances() {
  // Передній датчик
  frontDistance = measureDistance(TRIGGER_PIN_FRONT, ECHO_PIN_FRONT);
  delay(50);

  // Лівий датчик
  leftDistance = measureDistance(TRIGGER_PIN_LEFT, ECHO_PIN_LEFT);
  delay(50);

  // Правий датчик
  rightDistance = measureDistance(TRIGGER_PIN_RIGHT, ECHO_PIN_RIGHT);

  Serial.printf("Датчики - Front: %.1f см, Left: %.1f см, Right: %.1f см\n",
                frontDistance, leftDistance, rightDistance);
```

```
  // Відправляємо дані сенсорів на ESP32-CAM
  sendSensorDataToCAM();
}

// Функція для відправки даних сенсорів на ESP32-CAM
void sendSensorDataToCAM() {
  String sensorData = "SENSORS," + String(frontDistance, 1) + "," +
                      String(leftDistance, 1) + "," + String(rightDistance, 1);
  Serial1.println(sensorData);
  Serial.println("Відправлено на ESP32-CAM: " + sensorData);
}



// Функція для обертання моторів
void rotateMotor(int motorNumber, int motorDirection) {
  if (motorDirection == FORWARD) {
    digitalWrite(motorPins[motorNumber].pinIN1, HIGH);
    digitalWrite(motorPins[motorNumber].pinIN2, LOW);
  } else if (motorDirection == BACKWARD) {
    digitalWrite(motorPins[motorNumber].pinIN1, LOW);
    digitalWrite(motorPins[motorNumber].pinIN2, HIGH);
  } else {
    digitalWrite(motorPins[motorNumber].pinIN1, LOW);
    digitalWrite(motorPins[motorNumber].pinIN2, LOW);
  }

  if (motorDirection != STOP) {
    ledcWrite(motorPins[motorNumber].pwmSpeedChannel, currentSpeed);
  } else {
    ledcWrite(motorPins[motorNumber].pwmSpeedChannel, 0);
  }
}

// Функція для керування сервоприводами рульового керування
void setSteeringPosition(int position) {
  steeringServo1.write(position);
  steeringServo2.write(position);
  Serial.print("Встановлено позицію керуючих сервоприводів: ");
  Serial.println(position);
}

// Обробка руху робота
void processRobotMovement(int inputValue) {
  Serial.print("Команда руху: ");
  Serial.println(inputValue);

  switch (inputValue) {
    case UP:
      Serial.println("Рух вперед");
      rotateMotor(RIGHT_MOTOR, FORWARD);
      rotateMotor(LEFT_MOTOR, FORWARD);
      setSteeringPosition(90);
      break;
    case DOWN:
      Serial.println("Рух назад");
      rotateMotor(RIGHT_MOTOR, BACKWARD);
      rotateMotor(LEFT_MOTOR, BACKWARD);
```

```
      setSteeringPosition(90);
      break;
    case LEFT:
      Serial.println("Поворот ліворуч");
      rotateMotor(RIGHT_MOTOR, FORWARD);
      rotateMotor(LEFT_MOTOR, STOP);
      setSteeringPosition(45);
      break;
    case RIGHT:
      Serial.println("Поворот праворуч");
      rotateMotor(LEFT_MOTOR, FORWARD);
      rotateMotor(RIGHT_MOTOR, STOP);
      setSteeringPosition(135);
      break;
    case STOP:
    default:
      Serial.println("Зупинка");
      rotateMotor(RIGHT_MOTOR, STOP);
      rotateMotor(LEFT_MOTOR, STOP);
      setSteeringPosition(90);
      break;
  }
}

// Налаштування pin-режимів
void setUpPinModes() {
  Serial.println("Налаштування пінів...");

  // Налаштування пінів для ультразвукових датчиків
  pinMode(TRIGGER_PIN_FRONT, OUTPUT);
  pinMode(ECHO_PIN_FRONT, INPUT);
  pinMode(TRIGGER_PIN_LEFT, OUTPUT);
  pinMode(ECHO_PIN_LEFT, INPUT);
  pinMode(TRIGGER_PIN_RIGHT, OUTPUT);
  pinMode(ECHO_PIN_RIGHT, INPUT);

  // Початкове налаштування TRIG пінів
  digitalWrite(TRIGGER_PIN_FRONT, LOW);
  digitalWrite(TRIGGER_PIN_LEFT, LOW);
  digitalWrite(TRIGGER_PIN_RIGHT, LOW);

  // Налаштування серводвигунів
  panServo.attach(PAN_SERVO_PIN);
  panServo.write(90);

  tiltServo.attach(TILT_SERVO_PIN);
  tiltServo.write(0);

  steeringServo1.attach(STEERING_SERVO1_PIN);
  steeringServo2.attach(STEERING_SERVO2_PIN);
  steeringServo1.write(90);
  steeringServo2.write(90);

  // Налаштування моторів
  for (int i = 0; i < motorPins.size(); i++) {
    pinMode(motorPins[i].pinIN1, OUTPUT);
    pinMode(motorPins[i].pinIN2, OUTPUT);

    ledcSetup(motorPins[i].pwmSpeedChannel, PWMFreq, PWMResolution);
```

```
    ledcAttachPin(motorPins[i].pinEn, motorPins[i].pwmSpeedChannel);

    rotateMotor(i, STOP);
  }

  Serial.println("Налаштування завершено!");
}

void setup() {
  Serial.begin(115200);
  Serial.println("\n=== Ініціалізація надводного робота ===");
  Serial1.begin(115200, SERIAL_8N1, 16, 17);

  // Налаштування pin-режимів
  setUpPinModes();

  // Затримка для стабілізації
  delay(1000);

  // Виконуємо перші вимірювання
  measureAllDistances();

  Serial.println("=== Надводний робот готовий! ===");
}

void loop() {
  // Вимірювання відстаней через регулярні інтервали
  unsigned long currentTime = millis();
  if (currentTime - lastDistanceMeasurement > MEASUREMENT_INTERVAL) {
    measureAllDistances();
    lastDistanceMeasurement = currentTime;
  }

  // Перевірка наявності вхідних даних від ESP32-CAM
  if (Serial1.available()) {
    String dataString = Serial1.readStringUntil('\n');
    std::string myData = dataString.c_str();
    std::istringstream ss(myData);
    std::string key, value;

    std::getline(ss, key, ',');
    std::getline(ss, value, ',');

    Serial.printf("Від ESP32-CAM: %s = %s\n", key.c_str(), value.c_str());

    int valueInt = atoi(value.c_str());

    if (key == "MoveRobot") {
      processRobotMovement(valueInt);
    } else if (key == "Speed") {
      currentSpeed = valueInt;
      Serial.printf("Швидкість: %d\n", currentSpeed);
    } else if (key == "Pan") {
      panServo.write(valueInt);
      Serial.printf("Pan: %d\n", valueInt);
    } else if (key == "Tilt") {
      tiltServo.write(valueInt);
      Serial.printf("Tilt: %d\n", valueInt);
    }
```

```
  }

  // Передача команд з Serial на Serial1
  if (Serial.available()) {
    String command = Serial.readStringUntil('\n');
    command.trim();

    Serial1.println(command);
    Serial.println("Команда відправлена: " + command);
  }
}
```

ДОДАТОК Г

Демонстраційний матеріал

№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК.416616.026 ПЗ	Пояснювальна записка	А4, 84 с.
		Додаткові матеріали	
2		Апробація результатів	А4, 4 с.
		роботи	
3		Програмний код для	А4, 14 с.
		ESP32-CAM	
4		Програмний код для	А4, 6 с.
		ESP32 DevKit V1	
5		Демонстраційний матеріал	А4, 10 с.
		у вигляді презентації	

					ГЮИК.416616.026 ВД
Змін.	Арк.	Номер докум.	Підпис	Дата	
Розроб.		Котенко В. А.			Розроблення макету надводного робота для збору та обробки даних у реальному часі Відомість кваліфікаційної роботи
Перевір.		Бронніков А. І.			
Н.контр.		Демська Н. П.			
Затв.		Невлюдов І.Ш.			

Літера	Аркуш	Аркушів
Н	1	1

Кафедра КІТАР ХНУРЕ