

ДОДАТОК А

Код програми

```
import tkinter as tk
import socket
import cv2
from PIL import Image, ImageTk
import numpy as np
import threading
import RPi.GPIO as GPIO
import time

# GPIO піни для керування двигунами
GPIO.setmode(GPIO.BOARD)
left_motor_pins = [11, 13, 15] # GPIO11, GPIO13, GPIO15
right_motor_pins = [16, 18, 22] # GPIO16, GPIO18, GPIO22

# Піни для ультразвукового датчика HC-SR04
TRIG = 8
ECHO = 10

# Ініціалізація пінів GPIO
for pin in left_motor_pins + right_motor_pins:
    GPIO.setup(pin, GPIO.OUT)
    GPIO.output(pin, False)

GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

# Глобальні змінні для сокету та потоку відеопотоку
client_socket = None
video_thread = None
stop_video_thread = False
```

```
# Змінні для керування рухом
current_direction = "STOP" # Змінна для збереження поточного напрямку
руху робота
```

```
# Функції для керування роботом
```

```
def connect_to_pi(host, port):
```

```
    global client_socket
```

```
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
    try:
```

```
        client_socket.connect((host, port))
```

```
        print(f"Connected to {host}:{port}")
```

```
        return True
```

```
    except Exception as e:
```

```
        print(f"Connection failed: {str(e)}")
```

```
        return False
```

```
def send_command(command):
```

```
    try:
```

```
        client_socket.send(command.encode())
```

```
    except Exception as e:
```

```
        print(f"Error sending command: {e}")
```

```
def start_robot():
```

```
    send_command("START")
```

```
def stop_robot():
```

```
    send_command("STOP")
```

```
    stop_motors()
```

```
def forward():
    global current_direction
    send_command("FORWARD")
    current_direction = "FORWARD"
    drive_forward()

def backward():
    global current_direction
    send_command("BACKWARD")
    current_direction = "BACKWARD"
    drive_backward()

def turn_left():
    global current_direction
    send_command("LEFT")
    current_direction = "LEFT"
    turn_left_motors()

def turn_right():
    global current_direction
    send_command("RIGHT")
    current_direction = "RIGHT"
    turn_right_motors()

def stop_motors():
    for pin in left_motor_pins + right_motor_pins:
        GPIO.output(pin, False)

def drive_forward():
    GPIO.output(left_motor_pins[0], True)
```

```
GPIO.output(left_motor_pins[1], False)
GPIO.output(left_motor_pins[2], True)
```

```
GPIO.output(right_motor_pins[0], True)
GPIO.output(right_motor_pins[1], False)
GPIO.output(right_motor_pins[2], True)
```

```
def drive_backward():
```

```
    GPIO.output(left_motor_pins[0], False)
    GPIO.output(left_motor_pins[1], True)
    GPIO.output(left_motor_pins[2], False)
```

```
    GPIO.output(right_motor_pins[0], False)
    GPIO.output(right_motor_pins[1], True)
    GPIO.output(right_motor_pins[2], False)
```

```
def turn_left_motors():
```

```
    GPIO.output(left_motor_pins[0], False)
    GPIO.output(left_motor_pins[1], True)
    GPIO.output(left_motor_pins[2], True)
```

```
    GPIO.output(right_motor_pins[0], True)
    GPIO.output(right_motor_pins[1], False)
    GPIO.output(right_motor_pins[2], True)
```

```
def turn_right_motors():
```

```
    GPIO.output(left_motor_pins[0], True)
    GPIO.output(left_motor_pins[1], False)
    GPIO.output(left_motor_pins[2], True)
```

```
GPIO.output(right_motor_pins[0], False)
GPIO.output(right_motor_pins[1], True)
GPIO.output(right_motor_pins[2], True)

# Читання відстані від ультразвукового датчика
def read_distance():
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()

    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150
    distance = round(distance, 2)
    return distance

# Логіка обходу перешкод
def obstacle_avoidance():
    while True:
        distance = read_distance()
        print(f"Distance: {distance} cm")
        if distance < 30:
            print("Obstacle detected. Turning...")
            stop()
            time.sleep(1)
```

```
backward()
time.sleep(1)
stop()
time.sleep(1)
turn_right()
time.sleep(1)
stop()
else:
    if current_direction == "STOP":
        stop_motors()
    elif current_direction == "FORWARD":
        drive_forward()
    elif current_direction == "BACKWARD":
        drive_backward()
    elif current_direction == "LEFT":
        turn_left_motors()
    elif current_direction == "RIGHT":
        turn_right_motors()

# GUI setup
root = tk.Tk()
root.title("Robot Control Program")

# Connection settings
connection_frame = tk.Frame(root, pady=10)
connection_frame.pack()

host_label = tk.Label(connection_frame, text="Host (IP Address):")
host_label.grid(row=0, column=0, padx=5)
host_entry = tk.Entry(connection_frame, width=20)
```

```
host_entry.grid(row=0, column=1, padx=5)

port_label = tk.Label(connection_frame, text="Port:")
port_label.grid(row=0, column=2, padx=5)
port_entry = tk.Entry(connection_frame, width=10)
port_entry.grid(row=0, column=3, padx=5)

connect_button = tk.Button(connection_frame, text="Connect",
command=lambda: connect_to_pi(host_entry.get(), int(port_entry.get())))
connect_button.grid(row=0, column=4, padx=5)

# Video stream display
video_label = tk.Label(root)
video_label.pack(pady=10)

# Control buttons
control_frame = tk.Frame(root)
control_frame.pack()

start_button = tk.Button(control_frame, text="Start", command=start_robot)
start_button.grid(row=0, column=0, padx=10, pady=5)

stop_button = tk.Button(control_frame, text="Stop", command=stop_robot)
stop_button.grid(row=0, column=1, padx=10, pady=5)

forward_button = tk.Button(control_frame, text="Forward",
command=forward)
forward_button.grid(row=1, column=0, padx=10, pady=5)
```

```
backward_button = tk.Button(control_frame, text="Backward",
command=backward)
```

```
backward_button.grid(row=2, column=0, padx=10, pady=5)
```

```
left_button = tk.Button(control_frame, text="Left", command=turn_left)
```

```
left_button.grid(row=1, column=1, padx=10, pady=5)
```

```
right_button = tk.Button(control_frame, text="Right", command=turn_right)
```

```
right_button.grid(row=2, column=1, padx=10, pady=5)
```

```
# Function to update video stream
```

```
def update_video_stream():
```

```
    global stop_video_thread
```

```
    while not stop_video_thread:
```

```
        try:
```

```
            frame = receive_frame()
```

```
            if frame is not None:
```

```
                frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
                frame = Image.fromarray(frame)
```

```
                frame = ImageTk.PhotoImage(image=frame)
```

```
                video_label.config(image=frame)
```

```
                video_label.image = frame
```

```
            except Exception as e:
```

```
                print(f"Error updating video stream: {e}")
```

```
                break
```

```
    print("Video stream stopped.")
```

```
# Function to receive frame from Raspberry Pi
```

```
def receive_frame():
```

```
    try:
```

```

message = client_socket.recv(4096000) # Adjust buffer size as needed
if len(message) > 0:
    frame_data = np.frombuffer(message, dtype=np.uint8)
    frame = cv2.imdecode(frame_data, cv2.IMREAD_COLOR)
    return frame
else:
    return None
except Exception as e:
    print(f"Error receiving frame: {e}")
    return None

# Start video stream thread
def start_video_stream():
    global video_thread
    video_thread = threading.Thread(target=update_video_stream)
    video_thread.start()

# Start video stream
start_video_stream()

# Start GUI main loop
root.mainloop()

# Cleanup
stop_video_thread = True
if client_socket:
    client_socket.close()
GPIO.cleanup() # Завершення програми, очищення ресурсів GPIO

```

ДОДАТОК Б
Демонстраційний матеріал



Рисунок Б.1 – Тема дипломного проєкту



Рисунок Б.2 – Постанова завдання

Мобільний робот як об'єкт керування

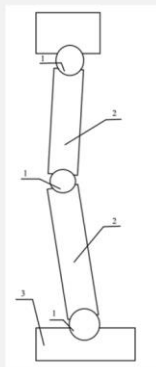


Схема ноги крокуючого робота

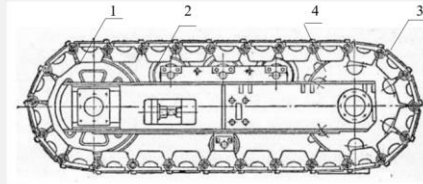
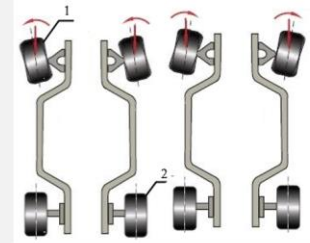


Схема гусеничної ланки



Загальний вигляд чотириколісного шасі

Рисунок Б.3 – Мобільний робот як об'єкт керування

Аналіз структури мобільних роботів

Структура мобільних роботів може включати різноманітні компоненти, призначені для виконання різних завдань та функцій. Основними компонентами структури мобільних роботів є:

- шасі;
- платформа керування;
- джерело живлення;
- модулі і датчики;
- актуатори;
- системи комунікації;
- навігаційні системи;
- системи комп'ютерного зору.

Рисунок Б.4 – Аналіз структури мобільних роботів

Аналіз і огляд сучасних методів дистанційного керування мобільним роботом

- ❑ Використання радіохвиль необхідне для передачі команд від пульта дистанційного керування до робота.
- ❑ Використання мобільної мережі дозволяє віддалено керувати роботом практично з будь-якого місця, де є зв'язок мобільного оператора.
- ❑ Використання супутникової мережі для віддаленого керування роботом забезпечує покриття практично всієї планети, але може мати високу затримку та високі витрати.
- ❑ Використання Інтернету для зв'язку між оператором та роботом через веб-інтерфейс або спеціалізовані додатки може бути досить універсальним, але потребує стабільного Інтернет-з'єднання.
- ❑ Бездротова мережа Wi-Fi використовується для зв'язку між роботом та оператором.

Рисунок Б.5 – Аналіз і огляд сучасних методів стану дистанційного керування мобільним роботом

Аналіз сучасного стану проблеми дистанційного керування мобільним роботом

- ❑ Сучасність відзначається значними досягненнями в галузі дистанційного керування мобільними роботами, проте існують проблеми, які потребують вирішення.
- ❑ Розвиток Wi-Fi, Bluetooth, мобільних мереж (3G/4G/5G) та супутникового зв'язку дозволяє ефективно віддалено керувати мобільними роботами на великій відстані.
- ❑ Деякі області можуть мати обмежене покриття мобільним зв'язком або Wi-Fi, що ускладнює дистанційне керування роботом.
- ❑ Вразливості в мережевому зв'язку можуть призвести до витоку інформації або зловживання контролем над роботом, що становить загрозу безпеці та конфіденційності.

Рисунок Б.6 – Аналіз сучасного стану проблеми дистанційного керування мобільним роботом

Організація систем дистанційного керування мобільними роботами на основі Internet of Things технологій



Рисунок Б.7 – Організація систем дистанційного керування мобільними роботами на основі Internet of Things технологій

Вибір компонентів мобільного робота



Рисунок Б.8 – Вибір компонентів мобільного робота

Обґрунтування вибору організації системи дистанційного керування

- ❑ Система керування мобільним роботом буде організована за допомогою методу P2P, що означає безпосереднє керування без посередницьких ланок, тобто без модуля обробки даних, тобто сервера.
- ❑ Плата Raspberry Pi 3 model B ідеально підходить для цих завдань і без проблем впорається з навантаженням керування і трансляції відео.
- ❑ Система керування включатиме в себе сервер, встановлений на мобільному роботі, і додаток, завдяки якому команди будуть передаватися на робота.
- ❑ Оскільки міні-комп'ютер вже обладнаний Wi-Fi, не потрібно окремо шукати, встановлювати і налаштувати інший модуль.
- ❑ Технологія Wi-Fi має значну дистанцію зв'язку і велику пропускну здатність, що дозволяє передавати будь-яку інформацію, зокрема відео з камери мобільного робота.

Рисунок Б.9 – Обґрунтування вибору організації системи дистанційного керування

Схема підключення компонентів мобільного робота

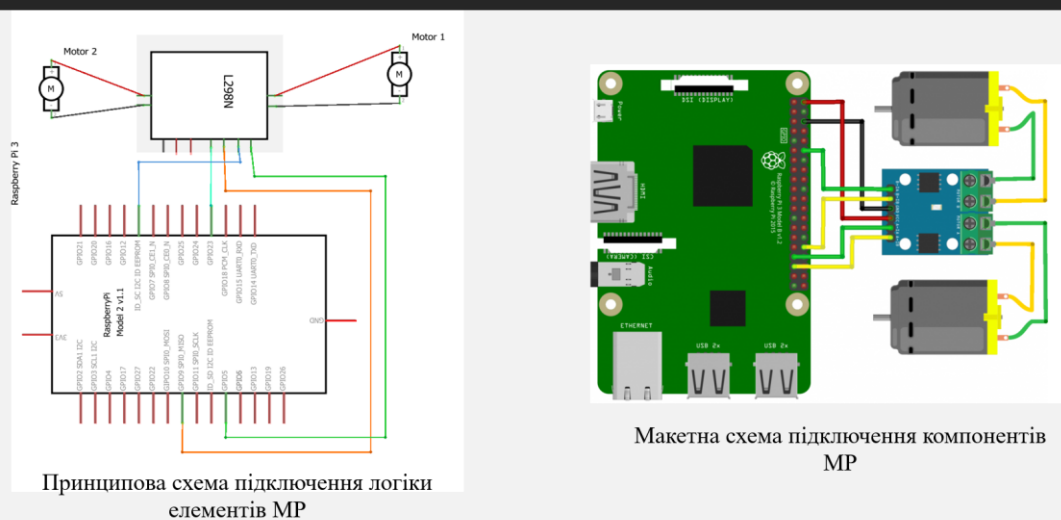


Рисунок Б.10 – Схема підключення компонентів мобільного робота

Схеми алгоритмів програм керування

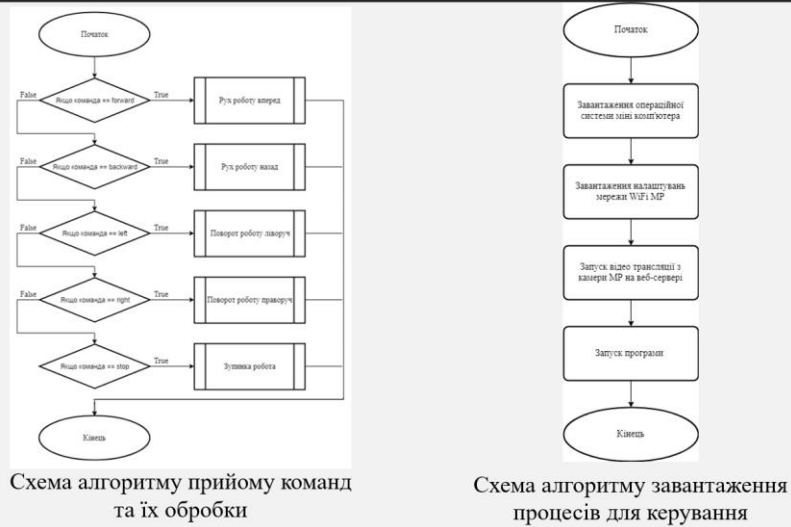


Рисунок Б.11 – Схеми алгоритмів програм керування

Розробка програми керування роботом

```
import socket
```

```
import RPi.GPIO as GPIO
```

```
import time
```

Підключення бібліотек



Алгоритм роботи за допомогою
ультразвукового датчика

```
GPIO.setup(IN1, GPIO.OUT)
GPIO.setup(IN2, GPIO.OUT)
GPIO.setup(IN3, GPIO.OUT)
GPIO.setup(IN4, GPIO.OUT)
GPIO.setup(ENA, GPIO.OUT)
GPIO.setup(ENB, GPIO.OUT)
```

Налаштування режимів
роботи портів
входу/виходу

Рисунок Б.12 – Розробка програми керування роботом

Розробка клієнтської програми

```
def connect_to_pi(host, port):
    global client_socket
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        client_socket.connect((host, port))
        print(f"Connected to {host}:{port}")
        return True
    except Exception as e:
        print(f"Connection failed: {str(e)}")
        return False
```

Створення з'єднання по сокету

```
def send_command(command):
    try:
        client_socket.send(command.encode())
    except Exception as e:
        print(f"Error sending command: {e}")
```

Створення відправки команд на МР

Рисунок Б.13 – Розробка клієнтської програми

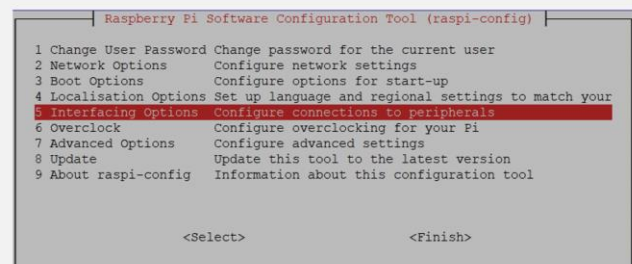
Встановлення необхідних пакетів та налаштувань

```
apt-get install hostapd dnsmasq git python-pip
```

```
pip install sockets times RPi.GPIO
```

```
python /home/pi/server_socket/robot_control.py
```

Завантаження утиліт та пакетів



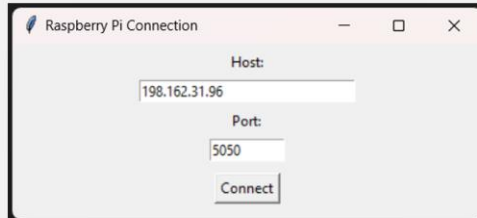
```
Raspberry Pi Software Configuration Tool (raspi-config)
1 Change User Password Change password for the current user
2 Network Options       Configure network settings
3 Boot Options         Configure options for start-up
4 Localisation Options Set up language and regional settings to match your
5 Interfacing Options  Configure connections to peripherals
6 Overclock            Configure overclocking for your Pi
7 Advanced Options    Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

<Select> <Finish>
```

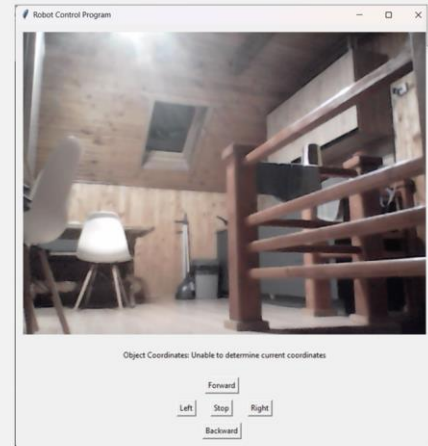
Вікно утиліти raspi-config

Рисунок Б.14 – Встановлення необхідних пакетів та налаштувань

Тестування програмного забезпечення



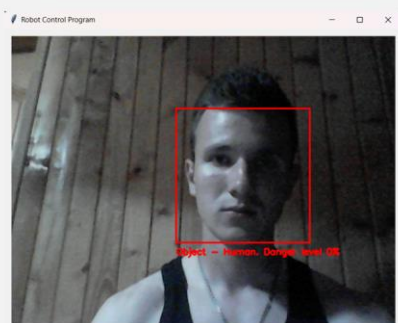
Інтерфейс програми підключення до МР



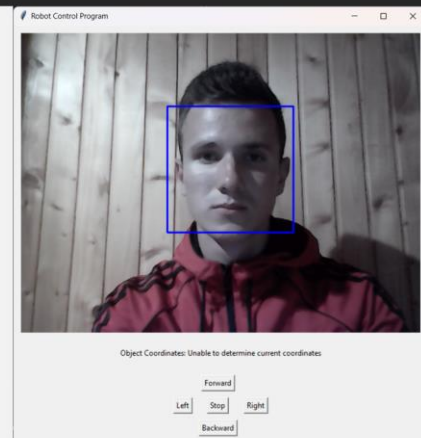
Інтерфейс програми керування МР

Рисунок Б.15 – Тестування програмного забезпечення

Тестування програмного забезпечення



Ідентифікація об'єкта та визначення рівня його небезпеки



Інтерфейс програми керування МР з системою розпізнавання обличчя

Рисунок Б.16 – Тестування програмного забезпечення

Аналіз розробленої системи

— Перевагами розробленої системи керування є:

- автономність;
- керування у реальному часі;
- можливість віддаленого керування;
- використання сучасної моделі для розпізнавання об'єктів підвищує точність і швидкість обробки даних;
- гнучкість налаштувань.

Проте, дана система керування має і певні недоліки:

- залежність від стабільної мережі;
- чутливість до навколишніх умов;
- енергозалежність.

Рисунок Б.17 – Аналіз розробленої системи

Висновки

— В результаті виконання роботи:

- проведено аналіз існуючих систем віддаленого керування мобільними роботами та аналіз аналогічних рішень;
- виконано підбір компонентів та реалізовано фізичний макет мобільного робота;
- розроблено алгоритми роботи системи віддаленого керування мобільним роботом, алгоритми та програму для мікроконтролера мобільного робота;
- проведено експерименти з випробування системи віддаленого керування гусеничним мобільним роботом.

Рекомендується проведення подальших досліджень, спрямованих на випробування системи автоматизації віддаленого управління мобільним роботом та провести оцінку ефективності в залежності від умов.

Рисунок Б.18 – Висновки

