ДОДАТОК А

Код логіки робота-асистента

```
import time
import speech_recognition as sr
import os
import socket
import random
import threading
import Adafruit_GPIO.SPI as SPI
import Adafruit_SSD1306
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont
from pushbullet import Pushbullet
from google import genai
from flask import Flask, request

current_web_command = None

client = genai.Client(api_key =
"AIzaSyCSXqjwXPMO6w4YloGNwmHyrdTSs6HKbv8")

RST = 24
DC = 23
SPI_PORT = 0
SPI_DEVICE = 0

disp = Adafruit_SSD1306.SSD1306_128_32(rst=RST)
disp.begin()
disp.clear()
disp.display()
width = disp.width
height = disp.height
image = Image.new('1', (width, height))

import RPi.GPIO as GPIO

from flask import Flask, request
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(23, GPIO.IN)
GPIO.setwarnings(False)
GPIO.setup(10, GPIO.OUT)
GPIO.setup(9, GPIO.IN)

app = Flask(__name__)
@app.route("/cmd")
def cmd():
    global current_web_command
    current_web_command = request.args.get("btn")
    return "OK"

def get_sonar_distance():
    GPIO.output(10, False)
    time.sleep(0.00001)
    GPIO.output(10, True)
    time.sleep(0.00001)
    GPIO.output(10, False)
    pulse_start = time.time()
    pulse_end = time.time()
    timeout = time.time() + 0.04

    while GPIO.input(9) == 0:
        pulse_start = time.time()
        if time.time() > timeout: return 1000

    while GPIO.input(9) == 1:
        pulse_end = time.time()
        if time.time() > timeout: return 1000

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

r = sr.Recognizer()

def check_internet_connection(host="8.8.8.8", port=53, timeout=3):
```

```
    try:
        socket.setdefaulttimeout(timeout)
        socket.socket(socket.AF_INET, socket.SOCK_STREAM).connect((host,
port))
        return True
    except Exception as ex:
        return False

if check_internet_connection():
    print("Connected")
    draw = ImageDraw.Draw(image)
    draw.rectangle((0, 0, width, height), outline=0, fill=0)
    draw.ellipse((15, 10, 55, 30), outline=1, fill=1)
    draw.ellipse((72, 10, 112, 30), outline=1, fill=1)

    draw.rectangle((15, 10, 55, 17), outline=0, fill=0)
    draw.rectangle((72, 10, 112, 17), outline=0, fill=0)

    draw.rectangle((15, 30, 55, 20), outline=0, fill=0)
    draw.rectangle((72, 30, 112, 20), outline=0, fill=0)

    draw.rectangle((15, 1, 55, 7), outline=0, fill=1)
    draw.rectangle((72, 1, 112, 7), outline=0, fill=1)
    disp.image(image)
    disp.display()

    time.sleep(0.0001)

    draw = ImageDraw.Draw(image)
    draw.rectangle((0, 0, width, height), outline=0, fill=0)
    draw.ellipse((15, 10, 55, 30), outline=1, fill=1)
    draw.ellipse((72, 10, 112, 30), outline=1, fill=1)

    draw.rectangle((15, 10, 55, 15), outline=0, fill=0)
    draw.rectangle((72, 10, 112, 17), outline=0, fill=0)

    draw.rectangle((15, 30, 55, 23), outline=0, fill=0)
    draw.rectangle((72, 30, 112, 20), outline=0, fill=0)

    draw.rectangle((15, 1, 55, 7), outline=0, fill=1)
    draw.rectangle((72, 1, 112, 7), outline=0, fill=1)
```

```
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0, 0, width, height), outline=0, fill=0)
draw.ellipse((15, 10, 55, 30), outline=1, fill=1)
draw.ellipse((72, 10, 112, 30), outline=1, fill=1)

draw.rectangle((15, 10, 55, 13), outline=0, fill=0)
draw.rectangle((72, 10, 112, 17), outline=0, fill=0)

draw.rectangle((15, 30, 55, 25), outline=0, fill=0)
draw.rectangle((72, 30, 112, 20), outline=0, fill=0)

draw.rectangle((15, 1, 55, 7), outline=0, fill=1)
draw.rectangle((72, 1, 112, 7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0, 0, width, height), outline=0, fill=0)
draw.ellipse((15, 10, 55, 30), outline=1, fill=1)
draw.ellipse((72, 10, 112, 30), outline=1, fill=1)

draw.rectangle((15, 10, 55, 11), outline=0, fill=0)
draw.rectangle((72, 10, 112, 15), outline=0, fill=0)

draw.rectangle((15, 30, 55, 27), outline=0, fill=0)
draw.rectangle((72, 30, 112, 23), outline=0, fill=0)

draw.rectangle((15, 1, 55, 7), outline=0, fill=1)
draw.rectangle((72, 1, 112, 7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)
```

```
draw = ImageDraw.Draw(image)
draw.rectangle((0, 0, width, height), outline=0, fill=0)
draw.ellipse((15, 10, 55, 30), outline=1, fill=1)
draw.ellipse((72, 10, 112, 30), outline=1, fill=1)

draw.rectangle((15, 10, 55, 9), outline=0, fill=0)
draw.rectangle((72, 10, 112, 13), outline=0, fill=0)

draw.rectangle((15, 30, 55, 29), outline=0, fill=0)
draw.rectangle((72, 30, 112, 25), outline=0, fill=0)

draw.rectangle((15, 1, 55, 7), outline=0, fill=1)
draw.rectangle((72, 1, 112, 7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0, 0, width, height), outline=0, fill=0)
draw.ellipse((15, 10, 55, 30), outline=1, fill=1)
draw.ellipse((72, 10, 112, 30), outline=1, fill=1)

draw.rectangle((15, 10, 55, 9), outline=0, fill=0)
draw.rectangle((72, 10, 112, 11), outline=0, fill=0)

draw.rectangle((15, 30, 55, 29), outline=0, fill=0)
draw.rectangle((72, 30, 112, 27), outline=0, fill=0)

draw.rectangle((15, 1, 55, 7), outline=0, fill=1)
draw.rectangle((72, 1, 112, 7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0, 0, width, height), outline=0, fill=0)
draw.ellipse((15, 10, 55, 30), outline=1, fill=1)
draw.ellipse((72, 10, 112, 30), outline=1, fill=1)
```

```
draw.rectangle((15, 10, 55, 9), outline=0, fill=0)
draw.rectangle((72, 10, 112, 9), outline=0, fill=0)

draw.rectangle((15, 30, 55, 29), outline=0, fill=0)
draw.rectangle((72, 30, 112, 29), outline=0, fill=0)

draw.rectangle((15, 1, 55, 7), outline=0, fill=1)
draw.rectangle((72, 1, 112, 7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(5)

draw = ImageDraw.Draw(image)
draw.rectangle((0, 0, width, height), outline=0, fill=0)
draw.ellipse((15, 10, 55, 30), outline=1, fill=1)
draw.ellipse((72, 10, 112, 30), outline=1, fill=1)

draw.rectangle((15, 10, 55, 9), outline=0, fill=0)
draw.rectangle((72, 10, 112, 9), outline=0, fill=0)

draw.rectangle((15, 30, 55, 29), outline=0, fill=0)
draw.rectangle((72, 30, 112, 29), outline=0, fill=0)

draw.rectangle((15, 1, 55, 7), outline=0, fill=1)
draw.rectangle((72, 1, 112, 7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0, 0, width, height), outline=0, fill=0)
draw.ellipse((15, 10, 55, 30), outline=1, fill=1)
draw.ellipse((72, 10, 112, 30), outline=1, fill=1)

draw.rectangle((15, 10, 55, 11), outline=0, fill=0)
draw.rectangle((72, 10, 112, 11), outline=0, fill=0)

draw.rectangle((15, 30, 55, 27), outline=0, fill=0)
draw.rectangle((72, 30, 112, 27), outline=0, fill=0)
```

```
draw.rectangle((15, 1, 55, 7), outline=0, fill=1)
draw.rectangle((72, 1, 112, 7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0, 0, width, height), outline=0, fill=0)
draw.ellipse((15, 10, 55, 30), outline=1, fill=1)
draw.ellipse((72, 10, 112, 30), outline=1, fill=1)

draw.rectangle((15, 10, 55, 13), outline=0, fill=0)
draw.rectangle((72, 10, 112, 13), outline=0, fill=0)

draw.rectangle((15, 30, 55, 25), outline=0, fill=0)
draw.rectangle((72, 30, 112, 25), outline=0, fill=0)

draw.rectangle((15, 1, 55, 7), outline=0, fill=1)
draw.rectangle((72, 1, 112, 7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0, 0, width, height), outline=0, fill=0)
draw.ellipse((15, 10, 55, 30), outline=1, fill=1)
draw.ellipse((72, 10, 112, 30), outline=1, fill=1)

draw.rectangle((15, 10, 55, 15), outline=0, fill=0)
draw.rectangle((72, 10, 112, 15), outline=0, fill=0)

draw.rectangle((15, 30, 55, 23), outline=0, fill=0)
draw.rectangle((72, 30, 112, 23), outline=0, fill=0)

draw.rectangle((15, 1, 55, 7), outline=0, fill=1)
draw.rectangle((72, 1, 112, 7), outline=0, fill=1)
disp.image(image)
disp.display()
```

```
    time.sleep(0.0001)

    draw = ImageDraw.Draw(image)
    draw.rectangle((0, 0, width, height), outline=0, fill=0)
    draw.ellipse((15, 10, 55, 30), outline=1, fill=1)
    draw.ellipse((72, 10, 112, 30), outline=1, fill=1)

    draw.rectangle((15, 10, 55, 17), outline=0, fill=0)
    draw.rectangle((72, 10, 112, 17), outline=0, fill=0)

    draw.rectangle((15, 30, 55, 20), outline=0, fill=0)
    draw.rectangle((72, 30, 112, 20), outline=0, fill=0)

    draw.rectangle((15, 1, 55, 7), outline=0, fill=1)
    draw.rectangle((72, 1, 112, 7), outline=0, fill=1)
    disp.image(image)
    disp.display()
else:
    print("Connection lost")
    draw = ImageDraw.Draw(image)
    draw.rectangle((0,0,width,height), outline=0, fill=0)
    draw.text((40, 8),    'NO WI-FI', fill=255)
    disp.image(image)
    disp.display()

def pressed_button():
    if GPIO.input(17) == True:
        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
        draw.rectangle((15,10,55,17), outline=0, fill=0)
        draw.rectangle((72,10,112,17), outline=0, fill=0)
        draw.rectangle((15,30,55,21), outline=0, fill=0)
        draw.rectangle((72,30,112,21), outline=0, fill=0)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()
```

```
time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,23), outline=0, fill=0)
draw.rectangle((72,30,112,23), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,25), outline=0, fill=0)
draw.rectangle((72,30,112,25), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,27), outline=0, fill=0)
draw.rectangle((72,30,112,27), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
```

```
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()

        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
        draw.rectangle((15,10,55,17), outline=0, fill=0)
        draw.rectangle((72,10,112,17), outline=0, fill=0)
        draw.rectangle((15,30,55,29), outline=0, fill=0)
        draw.rectangle((72,30,112,29), outline=0, fill=0)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()

        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
        draw.rectangle((15,10,55,17), outline=0, fill=0)
        draw.rectangle((72,10,112,17), outline=0, fill=0)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()
        print ('Button pressed')
        return wait_command()
    else:
        print ('Button don`t pressed')

def listen_for_hello_voxy():
    mic = sr.Microphone()
    with mic as source:
        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
```

```
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,21), outline=0, fill=0)
draw.rectangle((72,30,112,21), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,23), outline=0, fill=0)
draw.rectangle((72,30,112,23), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,25), outline=0, fill=0)
draw.rectangle((72,30,112,25), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()
```

```
time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,27), outline=0, fill=0)
draw.rectangle((72,30,112,27), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,29), outline=0, fill=0)
draw.rectangle((72,30,112,29), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
```

```
        disp.display()
        r.adjust_for_ambient_noise(source, duration=0.5)
        draw = ImageDraw.Draw(image)
        print('REC...')
        try:
            audio = r.listen(source, timeout=5, phrase_time_limit=5)  # Adjust the
timeout as needed
            text = r.recognize_google(audio, language='en-US')
            print(text)
            if 'hello' in text or 'hello voxy' in text or 'hello Vox' in text or 'hello boxing'
in text or 'hello voxin' in text or 'hello foxy' in text or 'hello boxer' in text or 'hello
voxing' in text or 'hello fox' in text or 'hello box' in text:
                return wait_command()
            else:
                draw = ImageDraw.Draw(image)
                draw.rectangle((0,0,width,height), outline=0, fill=0)
                draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
                draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
                draw.rectangle((15,10,55,17), outline=0, fill=0)
                draw.rectangle((72,10,112,17), outline=0, fill=0)
                draw.rectangle((15,1,55,7), outline=0, fill=1)
                draw.rectangle((72,1,112,7), outline=0, fill=1)
                disp.image(image)
                disp.display()

                time.sleep(0.0001)

                draw = ImageDraw.Draw(image)
                draw.rectangle((0,0,width,height), outline=0, fill=0)
                draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
                draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
                draw.rectangle((15,10,55,17), outline=0, fill=0)
                draw.rectangle((72,10,112,17), outline=0, fill=0)
                draw.rectangle((15,30,55,29), outline=0, fill=0)
                draw.rectangle((72,30,112,29), outline=0, fill=0)
                draw.rectangle((15,1,55,7), outline=0, fill=1)
                draw.rectangle((72,1,112,7), outline=0, fill=1)
                disp.image(image)
                disp.display()

                time.sleep(0.0001)
```

```
draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,27), outline=0, fill=0)
draw.rectangle((72,30,112,27), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,25), outline=0, fill=0)
draw.rectangle((72,30,112,25), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,23), outline=0, fill=0)
draw.rectangle((72,30,112,23), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
```

```
        disp.image(image)
        disp.display()

        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
        draw.rectangle((15,10,55,17), outline=0, fill=0)
        draw.rectangle((72,10,112,17), outline=0, fill=0)
        draw.rectangle((15,30,55,21), outline=0, fill=0)
        draw.rectangle((72,30,112,21), outline=0, fill=0)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()
        return False

except sr.WaitTimeoutError:
    draw = ImageDraw.Draw(image)
    draw.rectangle((0,0,width,height), outline=0, fill=0)
    draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
    draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
    draw.rectangle((15,10,55,17), outline=0, fill=0)
    draw.rectangle((72,10,112,17), outline=0, fill=0)
    draw.rectangle((15,1,55,7), outline=0, fill=1)
    draw.rectangle((72,1,112,7), outline=0, fill=1)
    disp.image(image)
    disp.display()

    time.sleep(0.0001)

    draw = ImageDraw.Draw(image)
    draw.rectangle((0,0,width,height), outline=0, fill=0)
    draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
    draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
    draw.rectangle((15,10,55,17), outline=0, fill=0)
    draw.rectangle((72,10,112,17), outline=0, fill=0)
    draw.rectangle((15,30,55,29), outline=0, fill=0)
    draw.rectangle((72,30,112,29), outline=0, fill=0)
```

```
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,27), outline=0, fill=0)
draw.rectangle((72,30,112,27), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,25), outline=0, fill=0)
draw.rectangle((72,30,112,25), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
```

```
        draw.rectangle((15,10,55,17), outline=0, fill=0)
        draw.rectangle((72,10,112,17), outline=0, fill=0)
        draw.rectangle((15,30,55,23), outline=0, fill=0)
        draw.rectangle((72,30,112,23), outline=0, fill=0)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()

        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
        draw.rectangle((15,10,55,17), outline=0, fill=0)
        draw.rectangle((72,10,112,17), outline=0, fill=0)
        draw.rectangle((15,30,55,21), outline=0, fill=0)
        draw.rectangle((72,30,112,21), outline=0, fill=0)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()
        print("You didn't say anything.")
        return False

    except sr.UnknownValueError:
        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
        draw.rectangle((15,10,55,17), outline=0, fill=0)
        draw.rectangle((72,10,112,17), outline=0, fill=0)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()

        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
```

```
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,29), outline=0, fill=0)
draw.rectangle((72,30,112,29), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,27), outline=0, fill=0)
draw.rectangle((72,30,112,27), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,25), outline=0, fill=0)
draw.rectangle((72,30,112,25), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()
```

```
        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
        draw.rectangle((15,10,55,17), outline=0, fill=0)
        draw.rectangle((72,10,112,17), outline=0, fill=0)
        draw.rectangle((15,30,55,23), outline=0, fill=0)
        draw.rectangle((72,30,112,23), outline=0, fill=0)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()

        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
        draw.rectangle((15,10,55,17), outline=0, fill=0)
        draw.rectangle((72,10,112,17), outline=0, fill=0)
        draw.rectangle((15,30,55,21), outline=0, fill=0)
        draw.rectangle((72,30,112,21), outline=0, fill=0)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()
        print("Sorry, I could not understand what you said.")
        return False

    except sr.RequestError:
        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
        draw.rectangle((15,10,55,17), outline=0, fill=0)
        draw.rectangle((72,10,112,17), outline=0, fill=0)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
```

```
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,29), outline=0, fill=0)
draw.rectangle((72,30,112,29), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
draw.rectangle((72,10,112,17), outline=0, fill=0)
draw.rectangle((15,30,55,27), outline=0, fill=0)
draw.rectangle((72,30,112,27), outline=0, fill=0)
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
draw.rectangle((15,10,55,17), outline=0, fill=0)
```

```
        draw.rectangle((72,10,112,17), outline=0, fill=0)
        draw.rectangle((15,30,55,25), outline=0, fill=0)
        draw.rectangle((72,30,112,25), outline=0, fill=0)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()

        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
        draw.rectangle((15,10,55,17), outline=0, fill=0)
        draw.rectangle((72,10,112,17), outline=0, fill=0)
        draw.rectangle((15,30,55,23), outline=0, fill=0)
        draw.rectangle((72,30,112,23), outline=0, fill=0)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()

        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
        draw.rectangle((15,10,55,17), outline=0, fill=0)
        draw.rectangle((72,10,112,17), outline=0, fill=0)
        draw.rectangle((15,30,55,21), outline=0, fill=0)
        draw.rectangle((72,30,112,21), outline=0, fill=0)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()
        print("Internet Connection Lost")
        return check_internet_connection()

def wait_command():
```

```
mic = sr.Microphone()
with mic as source:
    draw = ImageDraw.Draw(image)
    draw.rectangle((0,0,width,height), outline=0, fill=0)
    draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
    draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
    draw.rectangle((15,10,55,17), outline=0, fill=0)
    draw.rectangle((72,10,112,17), outline=0, fill=0)
    draw.rectangle((15,1,55,7), outline=0, fill=1)
    draw.rectangle((72,1,112,7), outline=0, fill=1)
    disp.image(image)
    disp.display()

    time.sleep(0.0001)

    draw = ImageDraw.Draw(image)
    draw.rectangle((0,0,width,height), outline=0, fill=0)
    draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
    draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
    draw.rectangle((15,10,55,15), outline=0, fill=0)
    draw.rectangle((72,10,112,15), outline=0, fill=0)
    draw.rectangle((15,1,55,7), outline=0, fill=1)
    draw.rectangle((72,1,112,7), outline=0, fill=1)
    disp.image(image)
    disp.display()

    time.sleep(0.0001)

    draw = ImageDraw.Draw(image)
    draw.rectangle((0,0,width,height), outline=0, fill=0)
    draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
    draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
    draw.rectangle((15,10,55,13), outline=0, fill=0)
    draw.rectangle((72,10,112,13), outline=0, fill=0)
    draw.rectangle((15,1,55,7), outline=0, fill=1)
    draw.rectangle((72,1,112,7), outline=0, fill=1)
    disp.image(image)
    disp.display()

    time.sleep(0.0001)
```

```
        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
        draw.rectangle((15,10,55,11), outline=0, fill=0)
        draw.rectangle((72,10,112,11), outline=0, fill=0)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()

        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
        draw.rectangle((15,10,55,9), outline=0, fill=0)
        draw.rectangle((72,10,112,9), outline=0, fill=0)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()

        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()
        r.adjust_for_ambient_noise(source, duration=0.5)
        draw = ImageDraw.Draw(image)
        print('REC 2...')
        try:
            audio = r.listen(source, timeout=5, phrase_time_limit=5)  # Adjust the
timeout as needed
            text = r.recognize_google(audio, language='en-US')
```

```
        print(text)
        if 'save event' in text or 'add event' in text or 'create event' in text:
            return save_event()
        elif 'check event' in text or 'show event' in text:
            return check_event()
        elif 'delete event' in text:
            return delete_event()
        elif 'I have a question' in text:
            return gemini()
        elif 'play' in text:
            return game()
        else:
            print('Error. unknow command')
            return listen_for_hello_voxy()
    except sr.WaitTimeoutError:
        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)

        draw.rectangle((15,10,55,9), outline=0, fill=0)
        draw.rectangle((72,10,112,9), outline=0, fill=0)

        draw.rectangle((15,30,55,29), outline=0, fill=0)
        draw.rectangle((72,30,112,29), outline=0, fill=0)

        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()

        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)

        draw.rectangle((15,10,55,11), outline=0, fill=0)
        draw.rectangle((72,10,112,11), outline=0, fill=0)
```

```
draw.rectangle((15,30,55,27), outline=0, fill=0)
draw.rectangle((72,30,112,27), outline=0, fill=0)

draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)

draw.rectangle((15,10,55,13), outline=0, fill=0)
draw.rectangle((72,10,112,13), outline=0, fill=0)

draw.rectangle((15,30,55,25), outline=0, fill=0)
draw.rectangle((72,30,112,25), outline=0, fill=0)

draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)

draw.rectangle((15,10,55,15), outline=0, fill=0)
draw.rectangle((72,10,112,15), outline=0, fill=0)

draw.rectangle((15,30,55,23), outline=0, fill=0)
draw.rectangle((72,30,112,23), outline=0, fill=0)

draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
```

```
        disp.image(image)
        disp.display()

        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)

        draw.rectangle((15,10,55,17), outline=0, fill=0)
        draw.rectangle((72,10,112,17), outline=0, fill=0)

        draw.rectangle((15,30,55,20), outline=0, fill=0)
        draw.rectangle((72,30,112,20), outline=0, fill=0)

        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()
        print("You didn't say anything.")
        return listen_for_hello_voxy()
    except sr.UnknownValueError:
        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)

        draw.rectangle((15,10,55,9), outline=0, fill=0)
        draw.rectangle((72,10,112,9), outline=0, fill=0)

        draw.rectangle((15,30,55,29), outline=0, fill=0)
        draw.rectangle((72,30,112,29), outline=0, fill=0)

        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()

        time.sleep(0.0001)
```

```
draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)

draw.rectangle((15,10,55,11), outline=0, fill=0)
draw.rectangle((72,10,112,11), outline=0, fill=0)

draw.rectangle((15,30,55,27), outline=0, fill=0)
draw.rectangle((72,30,112,27), outline=0, fill=0)

draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)

draw.rectangle((15,10,55,13), outline=0, fill=0)
draw.rectangle((72,10,112,13), outline=0, fill=0)

draw.rectangle((15,30,55,25), outline=0, fill=0)
draw.rectangle((72,30,112,25), outline=0, fill=0)

draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)
```

```
        draw.rectangle((15,10,55,15), outline=0, fill=0)
        draw.rectangle((72,10,112,15), outline=0, fill=0)

        draw.rectangle((15,30,55,23), outline=0, fill=0)
        draw.rectangle((72,30,112,23), outline=0, fill=0)

        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()

        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)

        draw.rectangle((15,10,55,17), outline=0, fill=0)
        draw.rectangle((72,10,112,17), outline=0, fill=0)

        draw.rectangle((15,30,55,20), outline=0, fill=0)
        draw.rectangle((72,30,112,20), outline=0, fill=0)

        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()
        print("Sorry, I could not understand what you said.")
        return listen_for_hello_voxy()
    except sr.RequestError:
        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)

        draw.rectangle((15,10,55,9), outline=0, fill=0)
        draw.rectangle((72,10,112,9), outline=0, fill=0)

        draw.rectangle((15,30,55,29), outline=0, fill=0)
        draw.rectangle((72,30,112,29), outline=0, fill=0)
```

```
draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)

draw.rectangle((15,10,55,11), outline=0, fill=0)
draw.rectangle((72,10,112,11), outline=0, fill=0)

draw.rectangle((15,30,55,27), outline=0, fill=0)
draw.rectangle((72,30,112,27), outline=0, fill=0)

draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()

time.sleep(0.0001)

draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)

draw.rectangle((15,10,55,13), outline=0, fill=0)
draw.rectangle((72,10,112,13), outline=0, fill=0)

draw.rectangle((15,30,55,25), outline=0, fill=0)
draw.rectangle((72,30,112,25), outline=0, fill=0)

draw.rectangle((15,1,55,7), outline=0, fill=1)
draw.rectangle((72,1,112,7), outline=0, fill=1)
disp.image(image)
disp.display()
```

```
        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)

        draw.rectangle((15,10,55,15), outline=0, fill=0)
        draw.rectangle((72,10,112,15), outline=0, fill=0)

        draw.rectangle((15,30,55,23), outline=0, fill=0)
        draw.rectangle((72,30,112,23), outline=0, fill=0)

        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()

        time.sleep(0.0001)

        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.ellipse((15, 10 , 55, 30), outline=1, fill=1)
        draw.ellipse((72, 10 , 112, 30), outline=1, fill=1)

        draw.rectangle((15,10,55,17), outline=0, fill=0)
        draw.rectangle((72,10,112,17), outline=0, fill=0)

        draw.rectangle((15,30,55,20), outline=0, fill=0)
        draw.rectangle((72,30,112,20), outline=0, fill=0)

        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
        disp.image(image)
        disp.display()
        print("Internet Connection Lost")
        return check_internet_connection()

def save_event():
    mic = sr.Microphone()
```

```
    with mic as source:
        draw = ImageDraw.Draw(image)
        draw.rectangle((15,10,112,30), outline=0, fill=0)

        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)

        draw.line((15, 15, 20, 10), fill=1)
        draw.line((20, 10, 30, 10), fill=1)
        draw.line((30, 10, 35, 15), fill=1)
        draw.line((27, 20, 35, 15), fill=1)
        draw.line((27, 20, 27, 25), fill=1)

        draw.line((92, 15, 97, 10), fill=1)
        draw.line((97, 10, 107, 10), fill=1)
        draw.line((107, 10, 112, 15), fill=1)
        draw.line((104, 20, 112, 15), fill=1)
        draw.line((104, 20, 104, 25), fill=1)

        disp.image(image)
        disp.display()
        r.adjust_for_ambient_noise(source, duration=0.5)
        print('Save event...')
        try:
            audio = r.listen(source, timeout=3, phrase_time_limit=3)  # Adjust the
timeout as needed
            text = r.recognize_google(audio, language='en-US')
            event = text
            with open("events.txt", "a") as file:
                file.write(f"{event}\n")
                print("Event saved")
                draw = ImageDraw.Draw(image)
                draw.rectangle((0,0,width,height), outline=0, fill=0)
                draw.text((20, 8),    'COMPLETE', fill=255)
                disp.image(image)
                disp.display()
                return listen_for_hello_voxy()
        except sr.WaitTimeoutError:
            draw = ImageDraw.Draw(image)
            draw.rectangle((15,10,112,30), outline=0, fill=0)
            draw.line((15, 18, 55, 18), fill=255)
```

```
            draw.line((72, 18, 112, 18), fill=255)
            disp.image(image)
            disp.display()
            print("You didn't say anything.")
            return
        except sr.UnknownValueError:
            draw = ImageDraw.Draw(image)
            draw.rectangle((15,10,112,30), outline=0, fill=0)
            draw.line((15, 18, 55, 18), fill=255)
            draw.line((72, 18, 112, 18), fill=255)
            disp.image(image)
            disp.display()
            print("Извините, не могу распознать речь. Пожалуйста, повторите.")
        except sr.RequestError as e:
            draw = ImageDraw.Draw(image)
            draw.rectangle((15,10,112,30), outline=0, fill=0)
            draw.line((15, 18, 55, 18), fill=255)
            draw.line((72, 18, 112, 18), fill=255)
            disp.image(image)
            disp.display()
            print(f"Ошибка сервиса распознавания речи; {e}")
        except sr.RequestError:
            draw = ImageDraw.Draw(image)
            draw.rectangle((15,10,112,30), outline=0, fill=0)
            draw.line((15, 18, 55, 18), fill=255)
            draw.line((72, 18, 112, 18), fill=255)
            disp.image(image)
            disp.display()
            print("Internet Connection Lost")
            return check_internet_connection()

def check_event():
    mic = sr.Microphone()
    events_found = []  # To store all events found

    with mic as source:
        draw = ImageDraw.Draw(image)
        draw.rectangle((15,10,112,30), outline=0, fill=0)

        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)
```

```
        draw.line((15, 15, 20, 10), fill=1)
        draw.line((20, 10, 30, 10), fill=1)
        draw.line((30, 10, 35, 15), fill=1)
        draw.line((27, 20, 35, 15), fill=1)
        draw.line((27, 20, 27, 25), fill=1)

        draw.line((92, 15, 97, 10), fill=1)
        draw.line((97, 10, 107, 10), fill=1)
        draw.line((107, 10, 112, 15), fill=1)
        draw.line((104, 20, 112, 15), fill=1)
        draw.line((104, 20, 104, 25), fill=1)

        disp.image(image)
        disp.display()
        r.adjust_for_ambient_noise(source, duration=0.5)
        print('Check event...')
        try:
            audio = r.listen(source, timeout=5, phrase_time_limit=5)  # Adjust the
timeout as needed
            text = r.recognize_google(audio, language='en-US')
            event = text
            with open("events.txt", "r") as file:
                for line in file:
                    if f"{event}" in line:
                        events_found.append(line.strip())  # Add the event to the list

            if events_found:
                draw = ImageDraw.Draw(image)
                draw.rectangle((0,0,width,height), outline=0, fill=0)
                draw.text((40, 8),'EVENT FOUND', fill=255)
                disp.image(image)
                disp.display()
                pb = Pushbullet("o.kkJE0qnfRXtDi8OxNWKHe9P2xSaBfERS")
                print(pb.devices)
                dev = pb.get_device('Xiaomi M2006C3LG')

                for event in events_found:  # Push each found event
                    push = dev.push_note("Event Notification", event)
                    print(event)
```

```
            return listen_for_hello_voxy()  # Return after processing all events

        else:
            draw = ImageDraw.Draw(image)
            draw.rectangle((0,0,width,height), outline=0, fill=0)
            draw.text((40, 8),'NO EVENT', fill=255)
            disp.image(image)
            disp.display()
            print('No event')
            return listen_for_hello_voxy()

    except sr.WaitTimeoutError:
        print("You didn't say anything.")
        draw = ImageDraw.Draw(image)
        draw.rectangle((15,10,112,30), outline=0, fill=0)
        draw.line((15, 18, 55, 18), fill=255)
        draw.line((72, 18, 112, 18), fill=255)
        disp.image(image)
        disp.display()
    except sr.UnknownValueError:
        draw = ImageDraw.Draw(image)
        draw.rectangle((15,10,112,30), outline=0, fill=0)
        draw.line((15, 18, 55, 18), fill=255)
        draw.line((72, 18, 112, 18), fill=255)
        disp.image(image)
        disp.display()
        print("Can`t recognize.")
    except sr.RequestError as e:
        draw = ImageDraw.Draw(image)
        draw.rectangle((15,10,112,30), outline=0, fill=0)
        draw.line((15, 18, 55, 18), fill=255)
        draw.line((72, 18, 112, 18), fill=255)
        disp.image(image)
        disp.display()
        print(f"Error of service {e}")
    except sr.RequestError:
        draw = ImageDraw.Draw(image)
        draw.rectangle((15,10,112,30), outline=0, fill=0)
        draw.line((15, 18, 55, 18), fill=255)
        draw.line((72, 18, 112, 18), fill=255)
        disp.image(image)
```

```
            disp.display()
            print("Internet Connection Lost")
            return check_internet_connection()

def delete_event():
    mic = sr.Microphone()

    with mic as source:
        draw = ImageDraw.Draw(image)
        draw.rectangle((15,10,112,30), outline=0, fill=0)

        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)

        draw.line((15, 15, 20, 10), fill=1)
        draw.line((20, 10, 30, 10), fill=1)
        draw.line((30, 10, 35, 15), fill=1)
        draw.line((27, 20, 35, 15), fill=1)
        draw.line((27, 20, 27, 25), fill=1)

        draw.line((92, 15, 97, 10), fill=1)
        draw.line((97, 10, 107, 10), fill=1)
        draw.line((107, 10, 112, 15), fill=1)
        draw.line((104, 20, 112, 15), fill=1)
        draw.line((104, 20, 104, 25), fill=1)

        disp.image(image)
        disp.display()
        r.adjust_for_ambient_noise(source, duration=0.5)
        print('Delete event...')
        try:
            audio = r.listen(source, timeout=5, phrase_time_limit=5)  # Adjust the
timeout as needed
            text = r.recognize_google(audio, language='en-US')
            print(text)
            event = text.strip()

            with open("events.txt", "r") as file:
                lines = file.readlines()

            with open("events.txt", "w") as file:
```

```
        for line in lines:
            if event in line:
                draw = ImageDraw.Draw(image)
                draw.rectangle((0,0,width,height), outline=0, fill=0)
                draw.text((40, 8),'SURE?', fill=255)
                disp.image(image)
                disp.display()
                print("Are you sure?")
                confirm_audio = r.listen(source, timeout=5)
                confirm_text = r.recognize_google(confirm_audio).lower()
                if "yes" in confirm_text:
                    draw = ImageDraw.Draw(image)
                    draw.rectangle((0,0,width,height), outline=0, fill=0)
                    draw.text((40, 8),'COMPLETE', fill=255)
                    disp.image(image)
                    disp.display()
                    print("complete")
                    return listen_for_hello_voxy()
                elif "no" in confirm_text or "cancel" in confirm_text:
                    draw = ImageDraw.Draw(image)
                    draw.rectangle((0,0,width,height), outline=0, fill=0)
                    draw.text((40, 8),'CANCELED', fill=255)
                    disp.image(image)
                    disp.display()
                    print("process cancelled")
                    file.write(line)
                    return listen_for_hello_voxy()# Write the line back to the file
            else:
                file.write(line)
                return listen_for_hello_voxy()
    except sr.WaitTimeoutError:
        print("You didn't say anything.")
        draw = ImageDraw.Draw(image)
        draw.rectangle((15,10,112,30), outline=0, fill=0)
        draw.line((15, 18, 55, 18), fill=255)
        draw.line((72, 18, 112, 18), fill=255)
        disp.image(image)
        disp.display()
    except sr.UnknownValueError:
        print("Can`t recognize")
        draw = ImageDraw.Draw(image)
```

```
            draw.rectangle((15,10,112,30), outline=0, fill=0)
            draw.line((15, 18, 55, 18), fill=255)
            draw.line((72, 18, 112, 18), fill=255)
            disp.image(image)
            disp.display()
        except sr.RequestError as e:
            draw = ImageDraw.Draw(image)
            draw.rectangle((15,10,112,30), outline=0, fill=0)
            draw.line((15, 18, 55, 18), fill=255)
            draw.line((72, 18, 112, 18), fill=255)
            disp.image(image)
            disp.display()
            print(f"Service error {e}")
        except sr.RequestError:
            draw = ImageDraw.Draw(image)
            draw.rectangle((15,10,112,30), outline=0, fill=0)
            draw.line((15, 18, 55, 18), fill=255)
            draw.line((72, 18, 112, 18), fill=255)
            disp.image(image)
            disp.display()
            print("Internet Connection Lost")
            return check_internet_connection()

def gemini():
    mic = sr.Microphone()

    with mic as source:
        draw = ImageDraw.Draw(image)
        draw.rectangle((15,10,112,30), outline=0, fill=0)

        draw.rectangle((15,1,55,7), outline=0, fill=1)
        draw.rectangle((72,1,112,7), outline=0, fill=1)

        draw.line((15, 15, 20, 10), fill=1)
        draw.line((20, 10, 30, 10), fill=1)
        draw.line((30, 10, 35, 15), fill=1)
        draw.line((27, 20, 35, 15), fill=1)
        draw.line((27, 20, 27, 25), fill=1)

        draw.line((92, 15, 97, 10), fill=1)
        draw.line((97, 10, 107, 10), fill=1)
```

```
        draw.line((107, 10, 112, 15), fill=1)
        draw.line((104, 20, 112, 15), fill=1)
        draw.line((104, 20, 104, 25), fill=1)

        disp.image(image)
        disp.display()
        r.adjust_for_ambient_noise(source, duration=0.5)
        try:
            audio = r.listen(source, timeout=3, phrase_time_limit=3)  # Adjust the
timeout as needed
            user_question = r.recognize_google(audio, language='en-US')
            print(f"You asked: {user_question}")
            prompt = f"{user_question} in a few words"

            response = client.models.generate_content(
                model="gemini-2.5-flash",
                contents=prompt,
                )
            gemini_answer = response.text
            print(f"Gemini Answer: {gemini_answer}")

            pb = Pushbullet("o.kkJE0qnfRXtDi8OxNWKHe9P2xSaBfERS")
            print(pb.devices)
            dev = pb.get_device('Xiaomi M2006C3LG')
            push = dev.push_note("Answer", gemini_answer)
            print("Answer sended")
            return listen_for_hello_voxy()

        except sr.WaitTimeoutError:
            draw = ImageDraw.Draw(image)
            draw.rectangle((15,10,112,30), outline=0, fill=0)
            draw.line((15, 18, 55, 18), fill=255)
            draw.line((72, 18, 112, 18), fill=255)
            disp.image(image)
            disp.display()
            print("You didn't say anything.")
            return
        except sr.UnknownValueError:
            draw = ImageDraw.Draw(image)
            draw.rectangle((15,10,112,30), outline=0, fill=0)
            draw.line((15, 18, 55, 18), fill=255)
```

```
            draw.line((72, 18, 112, 18), fill=255)
            disp.image(image)
            disp.display()
            print("can`t recornize.")
        except sr.RequestError as e:
            draw = ImageDraw.Draw(image)
            draw.rectangle((15,10,112,30), outline=0, fill=0)
            draw.line((15, 18, 55, 18), fill=255)
            draw.line((72, 18, 112, 18), fill=255)
            disp.image(image)
            disp.display()
            print(f"Service error {e}")
        except sr.RequestError:
            draw = ImageDraw.Draw(image)
            draw.rectangle((15,10,112,30), outline=0, fill=0)
            draw.line((15, 18, 55, 18), fill=255)
            draw.line((72, 18, 112, 18), fill=255)
            disp.image(image)
            disp.display()
            print("Internet Connection Lost")
            return check_internet_connection()


def game():
    global current_web_command
    print("--- START GAME MODE ---")

    draw = ImageDraw.Draw(image)
    draw.rectangle((0, 0, width, height), outline=0, fill=0)
    draw.text((30, 10), 'GAME MODE', fill=255)
    disp.image(image)
    disp.display()

    while True:

        if current_web_command is not None:
            cmd = current_web_command
            current_web_command = None

            if cmd == "stop":
                draw = ImageDraw.Draw(image)
```

```
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.text((40, 8),    'RETURN', fill=255)
        disp.image(image)
        disp.display()
        print("STOP")
        return listen_for_hello_voxy()

    elif cmd == "up":
        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.text((40, 8),    'UP', fill=255)
        disp.image(image)
        disp.display()
        print("UP")
    elif cmd == "down":
        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.text((40, 8),    'DOWN', fill=255)
        disp.image(image)
        disp.display()
        print("DOWN")
    elif cmd == "left":
        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.text((40, 8),    'LEFT', fill=255)
        disp.image(image)
        disp.display()
        print("LEFT")
    elif cmd == "right":
        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.text((40, 8),    'RIGHT', fill=255)
        disp.image(image)
        disp.display()
        print("RIGHT")

dist = get_sonar_distance()
if dist is not None and dist < 20:

    draw = ImageDraw.Draw(image)
    draw.rectangle((0, 0, width, height), outline=0, fill=0)
```

```
            draw.ellipse((15, 10, 55, 30), outline=1, fill=1)
            draw.ellipse((72, 10, 112, 30), outline=1, fill=1)

            draw.rectangle((15, 10, 55, 9), outline=0, fill=0)
            draw.rectangle((72, 10, 112, 9), outline=0, fill=0)

            draw.rectangle((15, 30, 55, 29), outline=0, fill=0)
            draw.rectangle((72, 30, 112, 29), outline=0, fill=0)

            draw.rectangle((15, 1, 55, 7), outline=0, fill=1)
            draw.rectangle((72, 1, 112, 7), outline=0, fill=1)
            disp.image(image)
            disp.display()
            print(f"Sonar: {dist} sm")

        if GPIO.input(23) == 0:

            draw = ImageDraw.Draw(image)
            draw.rectangle((0, 0, width, height), outline=0, fill=0)
            draw.ellipse((15, 10, 55, 30), outline=1, fill=1)
            draw.ellipse((72, 10, 112, 30), outline=1, fill=1)

            draw.rectangle((15, 10, 55, 17), outline=0, fill=0)
            draw.rectangle((72, 10, 112, 17), outline=0, fill=0)

            draw.rectangle((15, 30, 55, 20), outline=0, fill=0)
            draw.rectangle((72, 30, 112, 20), outline=0, fill=0)

            draw.rectangle((15, 1, 55, 7), outline=0, fill=1)
            draw.rectangle((72, 1, 112, 7), outline=0, fill=1)
            disp.image(image)
            disp.display()
            print("IR WORK")

        time.sleep(0.05)

if __name__== "__main__":
    try:
        flask_thread = threading.Thread(target=lambda: app.run(host="0.0.0.0",
port=5000, use_reloader=False))
        flask_thread.daemon = True
```

```
    flask_thread.start()
    print("Web work")

    time.sleep(1)

    while True:

        if check_internet_connection():

            if pressed_button() or listen_for_hello_voxy():
                pass
        else:
            time.sleep(5)

except KeyboardInterrupt:
    print("\nStop_programm")
    GPIO.cleanup()
```

ДОДАТОК Б

Апробація результатів кваліфікаційної 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 2)

[електронне видання]

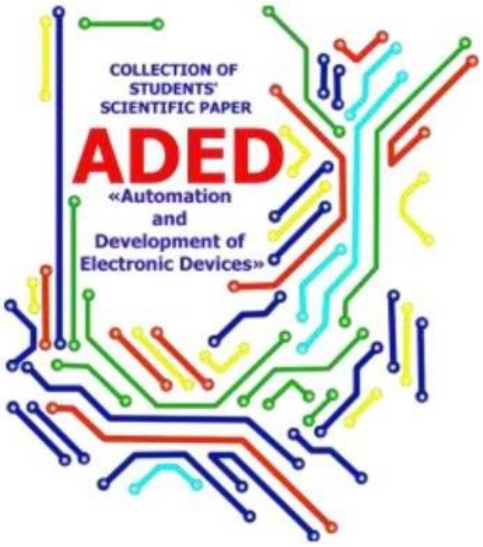
http://nure.ua/department/kafedra-komp-yuterno-integrovanih-tehnologiy-avtomatizatsiyi-ta-mehatroniki-kitam

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

http://kafea.kdu.edu.ua

Харків 2025

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(КІТАР)



ЗБІРНИК

студентських наукових статей

«Автоматизація та приладобудування»

«Automation and Development of Electronic Devices»

ADED-2025

(Випуск 2)

[електронне видання]

Харків 2025

Автоматизація та Приладобудування («Automation and Development of Electronic Devices» ADED-2025) [Електронний ресурс] : збірник студентських наукових статей / Харківський національний університет радіоелектроніки ; [редкол.: І.Ш. Невлюдов та ін.]. – Харків : ХНУРЕ, 2025. – Вип. 2. – 207с.

Collection of Students' Scientific Paper «Automation and Development Of Electronic Devices» ADED-2025 Part 2 (Key infrastructure 2025) - Kharkiv/ The Editorial.: Nevlyudov I.Sh. (head), that all. Kharkiv: Kind of Kharkiv National Universitety of Radio Elektronics [electronic edition], 2025. – 207p with.

Рекомендовано рішенням
Науково-технічної ради
Харківського національного
університету радіоелектроніки
протокол №6 від 29.11.2018

Рекомендовано рішенням Вченої ради
факультету Автоматики і комп'ютеризованих технологій
Харківського національного
університету радіоелектроніки
протокол № 4 від 21 листопада 2025

Збірник містить наукові статті здобувачів першого (бакалаврського), другого (магістерського) рівнів вищої освіти кафедри комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР) Харківського національного університету радіоелектроніки, кафедри Інформаційних технологій електронних засобів (ІТЕД) Запорізького національного технічного університету та кафедри Електронних апаратів (ЕА) Кременчуцького національного університету ім. М. Остроградського які навчаються за спеціальностями: 151 Автоматизація та комп'ютерно-інтегровані технології, 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка; 172 Телекомунікації та радіотехніка, 171 Електроніка та 163 Біомедична інженерія. Статті надані в авторській редакції.

УДК 004.8

АНАЛІЗ РОБОТИ З ШТУЧНИМИ ІНТЕЛЕКТАМИ

Д.С. Проценко
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
E-mail: dmytro.protsenko@nure.ua

***Анотація*:** У роботі основну увагу було приділено аналізу, порівнянню штучних інтелектів та підключенню безкоштовної моделі у свій проект. Було розглянуті популярні моделі ШІ, порівняно функціонал та популярність серед користувачів.

Ключові слова: штучний інтелект, ШІ, підключення.

COMPARISON OF METHODS OF INTERACTION WITH ASSISTANTS

D. Protsenko
Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, Nauky av., 14
E-mail: dmytro.protsenko@nure.ua

Annotation: The article focused on on the analyzing, comparing artificial intelligences, and integrating a free model into your project. Popular AI models were reviewed, their functionality and popularity among users were compared.

Key words: artificial intelligence, AI, connectivity.

На сьогоднішній день існує багато різновидів штучного інтелекту, які застосовуються у різних сферах. Наприклад найбільш популярний ChatGPT є чат-ботом, який працює на базі generative pre-trained transformer – тобто він навчається за допомогою великої кількості інформації зі статей, книг, інтернета та інших ресурсів. Він може аналізувати питання і шукати відповідь з різних джерел.

ChatGPT я і інші ШІ приймають та виводять інформацію за допомогою токенів. Це працює таким чином: коли користувач пише запит у чат, ШІ розбиває його на токени – фрагменти тексту, які можуть бути цілим словом, частиною слова або розділовим знаком. Таким чином, модель «розбиває» речення на токени, аналізує їх і потім і шукає відповідь. Знайшовши результат, він виводить його тільки в зворотньому порядку: токени аналізуються, будується речення і виводиться відповідь. Зазвичай кількість токенів є обмеженою, наприклад модель GPT-3.5 має пам'ять на 4 – 16 тисяч токенів, а модель GPT-4 має пам'ять до 128 тисяч токенів. Можемо помітити, що кількість токенів стало більшим у 8 раз. Це пов'язано із тим що GPT-4 — це новіша, потужніша архітектура, яка має більше пам'яті й ефективніше працює з токенами.

Через швидкий розвиток сфери штучного інтелекту, великі компанії почали створювати власні ШІ для виконання певних робіт через гнучкість: Grok, Gemini, Copilot, Claude, DeepSeek – такі самі чат боти як і раніше згаданий ChatGPT, але кожний такий чат має певні обмеження та налаштування на відповідь через закони країн, політичний стан і моральні принципи. ШІ від Nvidia створювався не для спілкування, а для роботи із рендером зображення у реальному часі, Midjourney – для створення зображень з тексту, Suna – для створення музики і пісень і багато інших. Також різні розробники створюють ШІ для обробки зображень з камери і звуку з мікрофону. Наприклад настільний робот EMO від LivingAI (рис. 1)[5] має вбудований нейромережевий процесор, який відповідає за аналіз емоцій користувача і його голос. Завдяки цим функціям, робот може підлаштовуватись під настрій власника, реагувати на тон і гучність голосу, бути більш гнучким у поведінці ніж звичайні запрограмовані роботи на певний перелік команд.

Рисунок 1 – Робот EMO від LivingAI

За аналізом сайту Artificial Analysis (сайт який займається порівнянням актуальних ШІ), станом на жовтень 2025, маємо результати зображені на графіках (рис.2–3)[2].

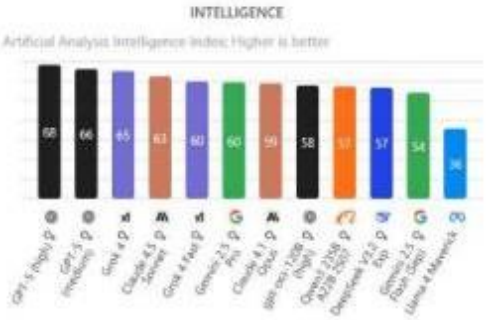


Рисунок 2 – Результати порівняння ШІ за інтелектуальним індексом

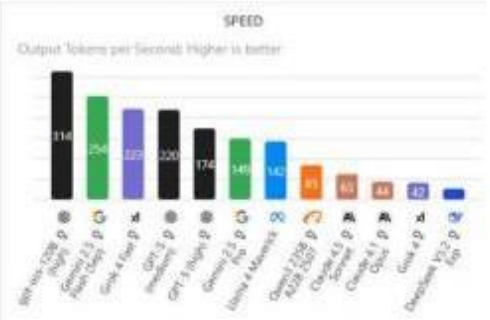


Рисунок 3 – Результати порівняння ШІ за швидкістю

Роблячи підсумки цих результатів, можемо зрузуміти, що за результатами порівняння рівня інтелектуального індексу (рис.2) 1 та 2 місце займає дві моделі ChatGPT 5 версії, а 3 місце – Grok 4. Проте, якщо брати аналіз швидкості генерації відповіді (рис.3), тоді маємо такі результати: 1 місце – ChatGPT gpt-oss-mini 120B, 2 місце – Gemini 2.5 Flash (модель, орієнтована на швидкі відповіді із збереженням їх високої якості), 3 місце – Grok 4 Fast (пришвидшена

модель Grok 4, як і Gemini 2.5 Flash). Також є багато інших показників чат-ботів які можна порівняти, але через їхній швидкий розвиток результати аналізу постійно змінюються.

Під час аналізу результатів порівнянь ШІ, згадувалась модель ChatGPT gpt-oss-mini 120B. Gpt-oss – серія відкритих моделей від компанії Open AI, яка дозволяє модифікувати і використовувати модель у власних роботах. Завдяки цьому, різні компанії можуть додавати штучний інтелект до своїх проектів у комерційних цілях. Проте швидкість обробки запитів може бути низькою через обмеження виводу токенів для економії.

Деякі компанії також надають можливість ознайомитись із штучним інтелектом у власних проектах. Однією з таких компаній є Google яка надає можливість згенерувати API ключ для під'єднання моделі штучного інтелекту до проекту. Якщо зайти на їх сторінку із налаштуваннями підключень ШІ до проектів, можна знайти усі необхідні інструкції. Наприклад, нам треба підключити Google Gemini, модель gemini-2.5-flash. По-перше, створюємо API ключ, вводимо назву проекту де буде застосовуватись згенерований ключ. В результаті отримаємо пункт, де написана уся інформація про створений ключ (рис. 4). Варто зазначити, що у розділі Quota tier написано, що ми маємо безкоштовний план користування. Це означає, що ми можемо тестувати модель у проекті, проте через велику кількість запитів, відповіді можуть обмежуватись по швидкості або по кількості.

Рисунок 4 – Інформація про створений API ключ

Наступним етапом підключення ШІ є встановлення необхідної бібліотеки. Google Gemini підтримує такі мови програмування як: Python, JavaScript, GO, Java, REST. Розберемо підключення на прикладі мови програмування Python. У терміналі на пристрої прописуємо бібліотеку для встановлення, а саме: pip install google-genai. Завдяки ній, ми зможемо працювати із моделлю у нашому проекті. Далі, необхідно під'єднати і налаштувати модель безпосередньо у проект. На сайті вже є приклад коду, в якому можемо задавати питання асистенту і отримувати відповідь, для цього просто необхідно додати наш згенерований API ключ (рис. 5).

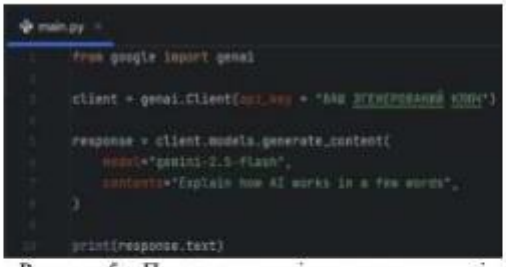


Рисунок 5 – Приклад коду підключення моделі

Завдяки цьому простому прикладу коду ми можемо перевірити, чи працює модель у нашому проекті. На запит "Explain how AI works in a few words", що перекладається як "Поясни як працює ШІ у декількох словах", наша програма даст коротку відповідь: "AI learns patterns from data to make decisions or predictions." - "Штучний інтелект вивчає закономірності з даних, щоб приймати рішення або робити прогнози." Але через гнучкість, нашу програму можна налаштувати як завгодно: зробити чат-бота у реальному часі або голосового асистента як у кіно, обмежити кількість токенів для економії і багато іншого.

Таким чином, компанії надають можливість ознайомитись із моделями ШІ не тільки як користувач, але і як розробник: під'єднати до власного проекту, розширити функціонал, додати

інтерактивності і т.д. До того ж, все більше корпорацій надає пробний безкоштовний доступ до своїх ШІ для ознайомлення, а вже потім користувачі приймають рішення, платити за них чи ні.

ВИСНОВКИ. За результатами аналізу ринку штучних інтелектів від різних компаній можна зрозуміти, що наразі існує великий вибір моделей, які націлені під різні задачі. Через швидкий розвиток сфери, порівняння швидкості, якості і енергоефективності може постійно змінюватись, через актуальність ринку і його постійне вдосконалення, тому що компанії розуміють на скільки це потужний інструмент. ШІ може не просто відповідати на запитання, він може генерувати медіа, прискорювати обробку даних, робити її більш злагодженою.

До того ж, варто зазначити, що деякі великі корпорації надають можливість безкоштовно ознайомитись із моделями і використати їх у власних проектах. Завдяки такому підходу, різні невеликі студії, розробники або ентузіасти можуть навчитися використовувати цей інструмент у власних розробках.

ЛІТЕРАТУРА

1. Google AI Studio. Google AI Studio. URL: https://aistudio.google.com/?project=gen-lang-client-0494708156.
2. Comparison of AI Models across Intelligence, Performance, Price | Artificial Analysis. AI Model & API Providers Analysis | Artificial Analysis. URL: https://artificialanalysis.ai/models.
3. The Complete Guide: How to Use ChatGPT API in Application. UI UX Design Agency for SaaS, Fintech & AI | Adam Fard UX Studio. URL: https://adamfard.com/blog/how-to-use-chatgpt-api#:~:text=Educational%20Tools:%20Educators%20and%20e,provide%20an%20engaging%20learning%20experience.
4. Gemini API | Google AI for Developers. Google AI for Developers. URL: https://ai.google.dev/gemini-api/docs.
5. EMO - LivingAI. LivingAI. URL: https://living.ai/emo/.
6. openai/gpt-oss-120b · Hugging Face. Hugging Face – The AI community building the future. URL: https://huggingface.co/openai/gpt-oss-120b?utm_source=chatgpt.com.
7. FaTPay. Understanding Token Limits in OpenAI's GPT Models. Medium. URL: https://medium.com/coinmonks/understanding-token-limits-in-openais-gpt-models-37fbe67c89f4.
8. GitHub - openai/gpt-oss: gpt-oss-120b and gpt-oss-20b are two open-weight language models by OpenAI. GitHub. URL: https://github.com/openai/gpt-oss.
9. Nevliudov, I., Yevsieiev, V., Maksymova, S., Gopejenko, V., & Kosenko, V. (2025). Development of mathematical support for adaptive control for the intelligent gripper of the collaborative robot manipulator. Advanced Information Systems, 9(3), 57-65.
10. Невлюдов, І., Клименко, О., Євсєєв, В., & Максимова, С. (2025). IMPROVEMENT OF THE ENCODING INFORMATION METHOD FOR PHARMACEUTICAL PRODUCTS QR-CODES DURING SORTING ON A ROBOTIC CONVEYOR LINE. Вісник Національного технічного університету «ХПІ». Серія: Технології в машинобудуванні, (1 (11)), 128-134.
11. Невлюдов, І., Євсєєв, В., Максимова, С., & Артюх, Р. (2025). Математична модель адаптивного ієрархічного високорівневого керування триланкового колаборативного робота-маніпулятора. Сучасний стан наукових досліджень та технологій в промисловості, (2 (32)), 58-68.
12. Demska, N., Yevsieiev, V., Maksymova, S., & Ababneh, J. (2025). DECISION-MAKING MODEL FOR CONTROLLING A COLLABORATIVE ROBOT-MANIPULATOR BASED ON THE SENSOR FUSION METHOD AND CNN APPROACH TO RULE FORMATION. Multidisciplinary Journal of Science and Technology, 5(6), 846-859.

Науковий керівник: Бронніков Артем Ігорович, к.т.н., доцент кафедри КІТАР.

ДОДАТОК В

Демонстраційний матеріал

Формат	Позиція	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість	Примітки
			Текстові документи		
А4	1	ГЮИК.ХХХХХХ.023 ПЗ	Пояснювальна	90	с.
			записка		
			Інші документи		
	2		Додаток А	40	с.
			Код логіки робота-		
			асистента		
			відстеження об’єктів		
	3		Додаток Б	9	с.
			Апробація		
			результатів		
			кваліфікаційної		
			роботи		
	4		Додаток В	14	с.
			Демонстраційний		
			матеріал		

Змін	Арк.	№ документу	Підпис	Дата	ГЮИК.ХХХХХХ.023 ПЗ			
Розробив		Проценко Д.Є.			Вдосконалення методів взаємодії з роботом-асистентом за допомогою сенсорної системи	Літера	Аркуш	Аркуш
Перевірив		Броніков А.І.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Норм. контроль		Демська Н.П.						
Затвердив		Невлюдов І.Ш.						