**Documentație Tehnică pentru Robotul Humanoid JOHN**

**Introducere**

Acest document detaliază construcția și funcționarea unui robot humanoid controlat de elevi. Robotul utilizează tehnologii avansate precum OpenAI, OpenCV, MediaPipe, Google Translate și pyttsx3 pentru a interacționa cu utilizatorii și mediul înconjurător. Controlul hardware este realizat prin intermediul unui Raspberry Pi, unui Arduino, un PCA6985 și 32 de servomotoare.

### Utilitatea Practică a Proiectului Robotului Humanoid

Proiectul robotului humanoid bazat pe tehnologii avansate precum OpenAI, OpenCV, MediaPipe, Google Translate și pyttsx3 are multiple aplicații practice și educaționale. Acestea includ:

#### 1. Educație și Învățare

* **Învățarea Programării:** Proiectul oferă elevilor o platformă practică pentru a învăța programare în Python și C/C++ (pentru Arduino). Elevii pot învăța despre structurile de control, funcțiile și bibliotecile externe.
* **Roboți și Automatizări:** Elevii pot învăța despre principiile de bază ale roboticii, inclusiv controlul motoarelor, senzorii și interacțiunea cu mediul înconjurător.
* **Inteligență Artificială:** Integrarea OpenAI și a tehnologiilor de recunoaștere a formelor și a vorbirii oferă o introducere în domeniul inteligenței artificiale și al învățării automate.

#### 2. Dezvoltare Tehnică și Inovare

* **Prototipare Rapidă:** Proiectul permite prototiparea rapidă a ideilor noi în robotică și inteligență artificială, fiind un punct de plecare pentru inovații ulterioare.
* **Integrarea Tehnologiilor:** Demonstrează cum diverse tehnologii pot fi integrate într-un sistem complex, pregătind elevii pentru proiecte mai avansate în viitor.

#### 3. Aplicații Practice

* **Asistență și Servicii:** Robotul poate fi folosit ca asistent vocal în medii casnice sau instituționale, ajutând la interacțiunea cu utilizatorii și oferind informații în timp real.
* **Educație Specială:** Poate fi folosit pentru a oferi suport educațional copiilor cu nevoi speciale, oferindu-le o interacțiune mai prietenoasă și accesibilă.
* **Monitorizare și Supraveghere:** Cu ajutorul camerei și al capacității de recunoaștere a formelor, robotul poate fi folosit pentru monitorizarea și supravegherea anumitor zone sau obiecte.

#### 4. Divertisment și Interacțiune

* **Interacțiuni Umane:** Robotul poate fi programat să interacționeze cu oamenii într-un mod amuzant și educativ, fiind o atracție în expoziții și evenimente.
* **Jocuri și Activități Educaționale:** Poate fi utilizat pentru a crea jocuri educaționale interactive, stimulând învățarea prin joacă.

#### 5. Cercetare și Dezvoltare

* **Cercetări în Robotică:** Proiectul poate servi ca platformă de cercetare pentru dezvoltarea de algoritmi noi în robotică și inteligență artificială.
* **Studii de Interacțiune Om-Robot:** Poate fi utilizat pentru studii privind interacțiunea om-robot, explorând moduri de îmbunătățire a experienței utilizatorilor.

### Concluzie

Proiectul robotului humanoid este o platformă multifuncțională care aduce beneficii educaționale, tehnice și practice. Prin implicarea în acest proiect, elevii își pot dezvolta abilități esențiale în programare, robotică și inteligență artificială, pregătindu-i pentru o carieră în domenii tehnologice avansate. Totodată, aplicațiile practice ale robotului deschid noi oportunități pentru inovare și cercetare, contribuind la progresul tehnologic.

**Componente Hardware**

**Raspberry Pi**

* Model: Raspberry Pi 5
* Funcție: Control principal, procesare imagine, recunoaștere vocală și sinteză vocală

**Arduino**

* Model: Arduino Uno
* Funcție: Controlul servomotoarelor pentru mișcările ploapelor

**Servomotoare**

* Număr: 32
* Model: MG996R, MG90S, TD-7120MG
* Funcție: Mișcări ale membrelor robotului (brațe, picioare, cap)

**Alte Componente**

* Cameră USB pentru Raspberry Pi
* Microfon USB pentru recunoaștere vocală
* Difuzor pentru sinteză vocală
* Breadboard și cabluri de conectare
* Alimentator pentru Raspberry Pi și Arduino

**Componente Software**

**Biblioteci și Pachete Python**

* OpenCV (opencv-python): Pentru procesare imagine și recunoaștere forme
* MediaPipe (mediapipe): Pentru recunoaștere și urmărire a mâinilor
* Google Translate (deep-translator): Pentru traducerea textului
* pyttsx3 (pyttsx3): Pentru sinteză vocală
* SpeechRecognition (speechrecognition): Pentru recunoaștere vocală
* OpenAI (openai): Pentru interacțiunea cu modelul de limbaj AI
* Adafruit ServoKit (adafruit-circuitpython-servokit): Pentru controlul servomotoarelor

**Instalarea Bibliotecilor**

Am utilizat pip pentru a instala bibliotecile necesare:

bash

pip install opencv-python mediapipe deep-translator pyttsx3 SpeechRecognition openai adafruit-circuitpython-servokit

**Configurarea Raspberry Pi și Arduino**

1. Am configurat Raspberry Pi OS pe Raspberry Pi 5.
2. Am conectat Raspberry Pi la internet și actualizat pachetele:

sudo apt-get update

sudo apt-get upgrade

**Schematica Electrică**

1. **Raspberry Pi Connections:**
   * Cameră USB conectată la un port USB
   * Microfon USB conectat la un port USB
   * Difuzor conectat la bluetooth audio
   * GPIO Pins pentru comunicare I2C cu PCA9685
2. **Arduino Connections:**
   * Pinii PWM conectați la servomotoare prin intermediul driverelor de servomotoare (PCA9685)
   * Conexiune I2C pentru comunicare cu Raspberry Pi

**Codul Python pentru Raspberry Pi**

**Importarea Bibliotecilor**

python

import speech\_recognition as sr

import pyttsx3

import openai

import numpy as np

import cv2

import pygame #pip install pygame

from pygame import mixer

import RPi.GPIO as GPIO

import mediapipe as mp

import time

import random

from deep\_translator import GoogleTranslator

from langdetect import detect

from adafruit\_servokit import ServoKit

mixer.init()

mixer.music.load("/home/cex/Downloads/WeRtheRobots.mp3")

kit = ServoKit(channels=16)

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

cap=cv2.VideoCapture(0)

mp\_drawing = mp.solutions.drawing\_utils

mp\_face = mp.solutions.face\_detection.FaceDetection(model\_selection=1,min\_detection\_confidence=0.5)

count=0

width=680

height=480

def obj\_data(img):

image\_input = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

results = mp\_face.process(image\_input)

if not results.detections:

print("noface")

else:

for detection in results.detections:

bbox = detection.location\_data.relative\_bounding\_box

# print(bbox)

x, y, w, h = int(bbox.xmin\*width), int(bbox.ymin \* height), int(bbox.width\*width),int(bbox.height\*height)

cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)

cx=int(x+x+w)//2

cy=int(y+y+h)//2

#cv2.cSearch

a=int(cx)//5

b=int(cy)//5

print("a",a)

print("b",b)

kit.servo[0].angle = a

kit.servo[14].angle=180-a

#kit.continuous\_servo[0].throttle = 1

kit.servo[1].angle=b

kit.servo[13].angle=b

#kit.continuous\_servo[1].throttle = 1

time.sleep(1)

#Initializing functiile pentru bibliotecile din text to speech

listening = True

engine = pyttsx3.init()

#Setam API-ul proiectului nostru cu OpenAI, si rolurile jucate de utilizatori si AI in acest proiect

openai.api\_key = "xxxx"

messages = [{"role": "system", "content": "Your name is John and give answers in 3 lines"}]

#Customizam vocea AI-ului

voices = engine.getProperty('voices')

rate = engine.getProperty('rate')

volume = engine.getProperty('volume')

#for voice in voices:

# engine.setProperty('voice',voice.id)

# engine.say('buna dimineata Soare')

#engine.runAndWait()

def get\_response(user\_input):

messages.append({"role": "user", "content": user\_input})

response = openai.ChatCompletion.create(

model = "gpt-3.5-turbo",

messages = messages

)

ChatGPT\_reply = response["choices"][0]["message"]["content"]

messages.append({"role": "assistant", "content": ChatGPT\_reply})

return ChatGPT\_reply

def moveServoSlowly(unghi, startAngle, endAngle, stepDelay):

if (startAngle < endAngle):

for i in range(startAngle, endAngle):

#unghi=range(0,15)

kit.servo[unghi].angle=i

time.sleep(stepDelay)

else:

for i in range(startAngle, endAngle, -1):

kit.servo[unghi].angle = i

time.sleep(stepDelay)

#servo 2 e dreaptaUmarX

#servo 3 e dreapta cot

#servo 4 e dreapta antebrat

#5 sunt degetele drepte fata

def moveServoSlowly2(unghi1,unghi2,unghi3,unghi4,unghi5, startAngle, endAngle, stepDelay):

if (startAngle < endAngle):

for i in range(startAngle, endAngle):

#unghi=range(0,15)

kit.servo[unghi1].angle=i

kit.servo[unghi2].angle=i

kit.servo[unghi3].angle=i

kit.servo[unghi4].angle=i

kit.servo[unghi5].angle=i

time.sleep(stepDelay)

else:

for i in range(startAngle, endAngle, -1):

kit.servo[unghi1].angle=i

kit.servo[unghi2].angle=i

kit.servo[unghi3].angle=i

kit.servo[unghi4].angle=i

kit.servo[unghi5].angle=i

time.sleep(stepDelay)

def salutDreapta():

moveServoSlowly(2, 0, 90, 0.02)

moveServoSlowly(3, 0, 90, 0.02)

moveServoSlowly(5, 150, 0, 0.02)

time.sleep(0.5)

print(1)

for i in range(0, 3):

moveServoSlowly(3, 0, 45, 0.02)

moveServoSlowly(3, 45, 0, 0.02)

print(2)

moveServoSlowly(5, 0, 150, 0.02)

time.sleep(1)

moveServoSlowly(3, 0, 150, 0.02)

moveServoSlowly(2, 90, 0, 0.02)

print(3)

#6 e umar stang

#7 e cot stang````

#8 e antebrat stang

#9 sunt degetele stangi fata

#uma r stang OY

def paharStanga():

moveServoSlowly2(1,13,15,15,15,0,90,0.02)

moveServoSlowly2(1,13,15,15,15,90,0,0.02)

moveServoSlowly(6, 25, 0, 0.02)

moveServoSlowly(7, 0, 90, 0.02)

moveServoSlowly(10, 0, 90, 0.02)

moveServoSlowly(8, 0, 45, 0.02)

moveServoSlowly2(9,11,15,12,15,180,0,0.02)

moveServoSlowly(10,90, 0, 0.02)

moveServoSlowly(8, 45, 22, 0.02)

#moveServoSlowly(6, 60, 150, 0.02)

time.sleep(1)

for i in range (0,10):

moveServoSlowly(8,0,150,0.005)

time.sleep(0.5)

time.sleep(0.5)

moveServoSlowly(6, 150, 0, 0.02)

#moveServoSlowly(7, 90, 0, 0.02)

moveServoSlowly(10, 0, 90, 0.02)

moveServoSlowly(8, 22, 45, 0.02)

moveServoSlowly2(9,11,12,15,15,0,180,0.02)

#time.sleep(1)

engine.say("here you go")

engine.runAndWait()

time.sleep(0.5)

moveServoSlowly(10, 90, 0, 0.02)

moveServoSlowly(7, 90, 0, 0.02)

#moveServoSlowly(8, 45, 0, 0.02)

def dansezi():

mixer.music.play()

time.sleep(2)

for i in range(0,5):

engine.say("i'm a robot")

moveServoSlowly(2, 0, 90, 0.005)

moveServoSlowly2(1,13,15,15,15, 0, 90, 0.005)

moveServoSlowly(3, 150, 0, 0.005)

moveServoSlowly(3, 0, 150, 0.005)

moveServoSlowly(10, 110, 180, 0.005)

engine.say("and i'm dancing")

moveServoSlowly2(1,13,15,15,15, 90, 180, 0.005)

moveServoSlowly(7, 0, 90, 0.005)

moveServoSlowly(7, 90, 0, 0.005)

moveServoSlowly(2, 90, 0, 0.005)

moveServoSlowly2(1,13,15,15,15, 150, 0, 0.005)

moveServoSlowly(10, 180, 110, 0.005)

engine.say("and i'm feeling good")

engine.runAndWait()

for i in range(0,10):

moveServoSlowly(2, 0, 90, 0.005)

moveServoSlowly2(1,13,15,15,15, 0, 90, 0.005)

moveServoSlowly(3, 150, 0, 0.005)

moveServoSlowly(3, 0, 150, 0.005)

moveServoSlowly(10, 0, 90, 0.005)

moveServoSlowly2(1,13,15,15,15, 90, 180, 0.005)

moveServoSlowly(7, 0, 90, 0.005)

moveServoSlowly(7, 90, 0, 0.005)

moveServoSlowly(2, 90, 0, 0.005)

moveServoSlowly2(1,13,15,15,15, 150, 0, 0.005)

moveServoSlowly(10, 90, 0, 0.005)

#for i in range(0,5):

# moveServoSlowly(3, , 0, 0.02)

def joci():

x=random.randint(0,2)

moveServoSlowly2(9,11,15,15,15,180,0,0.02)

moveServoSlowly2(15,15,12,15,15,0,180,0.02)

moveServoSlowly(10, 110, 70, 0.02)

moveServoSlowly(7, 90, 0, 0.02)

moveServoSlowly(8, 0, 140, 0.02)

moveServoSlowly(7, 0, 120, 0.02)

engine.say("rock")

engine.runAndWait()

moveServoSlowly(7, 120, 0, 0.005)

moveServoSlowly(7, 0, 120, 0.005)

engine.say("paper")

engine.runAndWait()

moveServoSlowly(7, 120, 0, 0.005)

moveServoSlowly(7, 0, 120, 0.005)

engine.say("scissor")

engine.runAndWait()

moveServoSlowly(7, 120, 0, 0.005)

engine.say("shoot")

engine.runAndWait()

moveServoSlowly(7, 0, 120, 0.005)

if x == 2:

moveServoSlowly(11, 0, 180, 0.005)

print("foarfeca")

engine.say("i chose scissor")

elif x==1:

moveServoSlowly2(9,11,12,15,15,0,180,0.01)

print("hartie")

engine.say("i chose paper")

else:

print("piatra")

engine.say("i chose rock")

time.sleep(1)

engine.runAndWait()

moveServoSlowly(10, 70, 110, 0.02)

moveServoSlowly2(9,11,15,15,12,0,180,0.01)

moveServoSlowly(7, 120, 0, 0.01)

moveServoSlowly(8, 140, 0, 0.02)

def multumesc():

moveServoSlowly2(15,15,15,15,7, 0, 90, 0.02)

moveServoSlowly2(15,15,15,15,6, 0, 20, 0.02)

moveServoSlowly(10, 90, 45, 0.02)

moveServoSlowly2(15,8,15,15,15, 0, 140, 0.02)

moveServoSlowly2(15,11,9,12,15, 180, 0, 0.02)

time.sleep(2)

moveServoSlowly2(15,11,9,12,15, 0, 180, 0.01)

moveServoSlowly2(15,15,15,15,7, 90, 0, 0.01)

moveServoSlowly(10, 45, 90, 0.02)

moveServoSlowly2(15,15,15,15,6, 20, 0, 0.01)

moveServoSlowly2(15,8,15,15,15, 140, 0, 0.01)

while listening:

ret,frame=cap.read()

count += 1

if count % 10 != 0:

continue

frame=cv2.resize(frame,(640,480))

frame=cv2.flip(frame,-1)

obj\_data(frame)

#cv2.imshow("FRAME",frame)

if cv2.waitKey(1)&0xFF==27:

engine.runAndWait()

break

with sr.Microphone() as source:

recognizer = sr.Recognizer()

recognizer.adjust\_for\_ambient\_noise(source)

recognizer.dynamic\_energy\_threshold = 3000

kit.servo[0].angle = 40

kit.servo[14].angle=140

kit.servo[1].angle = 110

kit.servo[13].angle = 70

try:

print("Listening...")

audio = recognizer.listen(source, timeout=5.0)

text = recognizer.recognize\_google(audio, language='ro-RO')

input\_language = detect(text)

print(text)

textul\_tradus=GoogleTranslator(source='ro',target='en').translate(text)

kit.servo[0].angle = 110

kit.servo[14].angle=70

kit.servo[1].angle = 10

kit.servo[13].angle=170

print(textul\_tradus)

#kit.servo[5].angle=20

#time.sleep(0.05)

#kit.servo[5].angle=100

if "john" in text.lower():

response\_from\_openai = get\_response(textul\_tradus)

engine.setProperty('rate',120)

engine.setProperty('volume', volume)

engine.setProperty('voice', 'american')

engine.say(response\_from\_openai,'John')

engine.runAndWait()

kit.servo[0].angle = 150

kit.servo[14].angle = 30

kit.servo[1].angle = 170

kit.servo[13].angle = 10

if "salut" in text.lower():

print("salutMANA")

salutDreapta()

if "shake" in text.lower():

print("aduPahar")

paharStanga()

if "dansezi" in text.lower():

print("danez")

dansezi()

if "mulțumesc" in text.lower():

engine.say("you are welcome")

print("multumesc")

engine.runAndWait()

multumesc()

if "joci" in text.lower():

engine.say("sure, I'd like to play. I am always up for a challenge")

print("joaca")

engine.runAndWait()

joci()

else:

print("Didn't recognize 'john'.")

except sr.UnknownValueError:

print("Didn't recognize anything.")

cap.release()

cv2.destroyAllWindows()

**Codul pentru Arduino**

**COD PENTRU GESTICULARE ROBOT**

**#**

include <Wire.h>

#include <Adafruit\_PWMServoDriver.h>

// Definirea canalelor pentru servomotoare

const int channelServoStangaUmarX = 0;

const int channelServoStangaUmarY = 1;

const int channelServoDreaptaUmarX = 2;

const int channelServoDreaptaUmarY = 3;

const int channelServoStangaCot = 4;

const int channelServoDreaptaCot = 5;

const int channelServoStangaAntebrat = 6;

const int channelServoDreaptaAntebrat = 7;

const int channelServoStangaDeget1 = 8;

const int channelServoStangaDeget2 = 9;

const int channelServoStangaDeget3 = 10;

const int channelServoStangaDeget4 = 11;

const int channelServoStangaDeget5 = 12;

// Pinul de intrare pentru modulul de detecție a sunetului

const int soundSensorPin = 2;

// Crearea obiectului pentru PCA9685

Adafruit\_PWMServoDriver pwm = Adafruit\_PWMServoDriver();

void setup() {

// Inițializare comunicare I2C

Wire.begin();

// Inițializare PCA9685

pwm.begin();

pwm.setPWMFreq(60); // Setare frecvență la 60 Hz pentru servomotoare

// Inițializare pin pentru modulul de detecție a sunetului

pinMode(soundSensorPin, INPUT);

// Inițializare seminte pentru funcția random

randomSeed(analogRead(0));

}

void loop() {

// Citirea valorii de la modulul de detecție a sunetului

int soundDetected = digitalRead(soundSensorPin);

// Dacă se detectează sunet

if (soundDetected == HIGH) {

// Exemplu de mișcări de gesticulare

gesticulate();

}

}

// Funcție pentru a seta unghiul unui servomotor

void moveServo(int channel, int angle) {

int pulseLength = map(angle, 0, 180, 150, 600); // Conversie unghi în puls

pwm.setPWM(channel, 0, pulseLength);

}

// Funcție pentru a seta unghiul unui servomotor lent

void moveServoSlowly(int channel, int startAngle, int endAngle, int stepDelay) {

if (startAngle < endAngle) {

for (int angle = startAngle; angle <= endAngle; angle++) {

moveServo(channel, angle);

delay(stepDelay);

}

} else {

for (int angle = startAngle; angle >= endAngle; angle--) {

moveServo(channel, angle);

delay(stepDelay);

}

}

}

// Funcție pentru mișcări de gesticulare

void gesticulate() {

// Mișcare 1

moveServoSlowly(channelServoStangaUmarX, 0, 45, 20);

moveServoSlowly(channelServoDreaptaUmarX, 0, 135, 20);

delay(500); // Așteaptă 0.5 secunde

// Mișcare 2

moveServoSlowly(channelServoStangaCot, 0, 90, 20);

moveServoSlowly(channelServoDreaptaCot, 0, 90, 20);

delay(500); // Așteaptă 0.5 secunde

// Mișcare 3

moveServoSlowly(channelServoStangaAntebrat, 0, 45, 20);

moveServoSlowly(channelServoDreaptaAntebrat, 0, 135, 20);

delay(500); // Așteaptă 0.5 secunde

// Mișcări aleatorii ale degetelor

moveServoSlowly(channelServoStangaDeget1, 0, random(0, 180), 20);

moveServoSlowly(channelServoStangaDeget2, 0, random(0, 180), 20);

moveServoSlowly(channelServoStangaDeget3, 0, random(0, 180), 20);

moveServoSlowly(channelServoStangaDeget4, 0, random(0, 180), 20);

moveServoSlowly(channelServoStangaDeget5, 0, random(0, 180), 20);

delay(500); // Așteaptă 0.5 secunde

// Resetare la poziția inițială

moveServoSlowly(channelServoStangaUmarX, 45, 0, 20);

moveServoSlowly(channelServoDreaptaUmarX, 135, 0, 20);

moveServoSlowly(channelServoStangaCot, 90, 0, 20);

moveServoSlowly(channelServoDreaptaCot, 90, 0, 20);

moveServoSlowly(channelServoStangaAntebrat, 45, 0, 20);

moveServoSlowly(channelServoDreaptaAntebrat, 135, 0, 20);

moveServoSlowly(channelServoStangaDeget1, 180, 0, 20);

moveServoSlowly(channelServoStangaDeget2, 180, 0, 20);

moveServoSlowly(channelServoStangaDeget3, 180, 0, 20);

moveServoSlowly(channelServoStangaDeget4, 180, 0, 20);

moveServoSlowly(channelServoStangaDeget5, 180, 0, 20);

delay(500); // Așteaptă 0.5 secunde}

**Concluzie**

Această documentație acoperă construcția și funcționarea unui robot humanoid care utilizează tehnologii avansate pentru a interacționa cu mediul înconjurător și utilizatorii săi. Utilizarea combinată a Raspberry Pi, Arduino, OpenAI, OpenCV, MediaPipe și alte tehnologii permite elevilor să învețe și să aplice concepte complexe într-un mod practic și interactiv.

