https://doi.org/10.59271/s45078.023.2301.11

Contents lists available at www.iusrj.org
International Uni-Scientific Research Journal
Journal homepage: www.iusrj.org

Circuit and Systems

Enhancing Accessibility and Independence of Visually Impaired Individuals through AI, ML and IoT: The Development of a Smart Robot Assistant.

Ahmed Adel, Ahmed Goda, Ahmed Essam, Mostafa Sadek, Mohammed K. Salama

Article Info

Article history:

Received: 01- 06 -2023

Accepted:13 – 06- 2023

doai202306012301

Available

Vol. 4 (11) 61-76

16th Nov 2023

Keywords:

smart robot, visually impaired

AI, ML, and IoT

Abstract

This paper presents the development of a smart robot assistant designed to enhance accessibility and independence for visually impaired individuals by leveraging Artificial Intelligence (AI), Machine Learning (ML), and Internet of Things (IoT) technologies. The system incorporates various sensors, including cameras, microphones, and distance sensors, with Raspberry Pi 4 and Nvidia RTX 3050 ti hardware to provide features such as voice recognition, obstacle detection, and navigation.

The software is programmed using Python version 3.9.16 and utilizes important libraries such as Tensorflow, OpenCV, and PyAudio. Our study shows that the smart robot assistant can offer numerous benefits, including increased mobility, safety, and independence, by recognizing and responding to voice commands, identifying obstacles and avoiding collisions, and providing audio feedback on its location and surroundings.

However, successful adoption requires addressing several challenges, including improving the accuracy and reliability of obstacle detection, ensuring privacy and security, and reducing costs. The propose strategies to overcome these challenges, such as leveraging AI and ML technologies, collaborating with stakeholders, and promoting regulatory frameworks.

In conclusion, this paper highlights the potential of AI, ML, and IoT technologies in developing smart robots to enhance the accessibility and independence of visually impaired individuals. By addressing the aforementioned challenges and incorporating user feedback, these systems have the potential to significantly improve the quality of life for this population.

Introduction

Visually impaired individuals face significant challenges in navigating and accessing the environment around them. To address this issue, we present the development of a smart robot assistant utilizing advanced Artificial Intelligence (AI), Machine Learning (ML), and Internet of Things (IoT) technologies to enhance accessibility and independence for visually impaired individuals. This paper aims to highlight the potential of smart robots in addressing the unique needs and challenges faced by this population.

The smart robot assistant is designed to provide a range of features such as voice recognition, obstacle detection, and navigation, utilizing various sensors including cameras, microphones, and distance sensors. Raspberry Pi 4 and Nvidia RTX 3050 ti hardware are used to process the data, while Python version 3.9.16 is utilized for software programming, integrating important libraries such as Tensorflow, OpenCV, and PyAudio.

Unlike traditional smart home devices aimed at improving convenience and efficiency, the smart robot assistant's primary goal is to enhance accessibility and independence for visually

Corresponding author
M. K. Salama
Assistant Professor. Faculty of Engineering CIC

impaired individuals. By leveraging the aforementioned technologies, it is capable of recognizing and responding to voice commands, identifying obstacles and avoiding collisions, and providing audio feedback on its location and surroundings.

The development of a smart robot assistant for Visually Impaired highlights the potential of AI, ML, and IoT technologies in enhancing accessibility and independence for visually impaired individuals. These smart robots can significantly improve the quality of life for vulnerable populations by addressing their unique needs and challenges.

Smart robots for smart homes serve as a centralized platform that allows homeowners to control and manage various smart devices, including lighting, heating/cooling systems, security systems, appliances, and more. Using a user-friendly interface, such as a mobile app or voice control, smart robots eliminate the need for manual device control, making it easier and more convenient for homeowners to manage their smart home.

Automating routine tasks is a key benefit of smart robots for smart homes. The smart robot can turn off lights, adjust temperature, and set the security system based on the homeowner's behavior. This automation streamlines the smart home experience, freeing up homeowners' time and energy for other activities.

Smart robots for smart homes also play an essential role in monitoring the home environment for potential security threats, such as intrusions or fires. When a potential threat is detected, the smart robot can alert the appropriate authorities or the homeowner, providing a layer of security and peace of mind. This feature is especially useful for homeowners who are frequently away from home, allowing them to monitor their home environment even when they are not there.
The development of a smart robot assistant using AI, ML, and IoT technologies has the potential to enhance accessibility and independence for visually impaired individuals. In addition to security, smart robots for smart homes can improve the efficiency of smart devices by monitoring their performance and making adjustments to ensure optimal functioning. They also have the ability to learn and adapt to the homeowner's habits and preferences over time, resulting in a more personalized and efficient smart home experience.

Smart robots for smart homes can be seamlessly integrated with other smart home technologies, such as smart assistants, providing homeowners with multiple options for controlling their smart devices. This integration allows for a convenient, efficient, and secure living experience.

Problem statement

Visually impaired individuals face numerous challenges in their daily lives. They may struggle with mobility and navigation, which can limit their independence and make it difficult to complete tasks such as shopping and cooking. Communicating with others can also be a challenge, and they may have trouble identifying objects and people in their surroundings. Accessing educational resources and opportunities is another hurdle, with limited resources and tools available to help them study and learn effectively. Ultimately, these challenges can impact their overall quality of life, making it essential to find innovative solutions that address their needs and enhance their independence and wellbeing.

Navigation: The smart robot can provide assistance to visually impaired individuals in navigating their environment, both indoors and outdoors. It can guide them through unfamiliar places, alert them of obstacles, and help them avoid potential hazards.

Object Recognition: Visually impaired individuals often struggle with identifying objects in their surroundings. A smart robot equipped with object recognition technology can help identify items such as household objects, food items, and even people to aid in social interactions.

Communication: Communication is a challenge for many visually impaired individuals. A smart robot capable of voice commands and responses can assist in making phone calls, sending messages, and accessing information online. The robot can also serve as a companion and engage in conversations with the user.

Independence: The smart robot can help visually impaired individuals become more independent by providing them with the necessary assistance to complete tasks on their own. This includes shopping, cooking, and managing daily routines.

Safety: The smart robot can enhance the safety of visually impaired individuals by monitoring their environment and alerting them to potential hazards such as open windows or doors. It can also alert emergency services in case of an accident or illness.

Education: The smart robot can help visually impaired individuals in their educational pursuits by providing them with access to digital books, documents, and other learning materials. The robot can also assist in note-taking and organizing study materials.

Assistive Technology: The smart robot can be integrated with assistive technology such as screen readers, magnifiers, and braille displays, to make it easier for visually impaired individuals to study and learn.

Tutoring: The smart robot can serve as a virtual tutor, providing personalized assistance to visually impaired students in various subjects. The robot can adapt to the student's learning style and pace, making studying more efficient and effective.

Accessibility: The smart robot can help visually impaired individuals access educational resources that may not be

readily available to them due to accessibility issues. This includes online courses, workshops, and conferences.

Overall, a smart robot for visually impaired individuals has the potential to improve their quality of life by addressing several challenges they face on a daily basis, including mobility, communication, and education.

Methodology

The concept of a methodology refers to a structured and standardized approach to problem-solving that helps individuals and teams plan, organize, and execute projects and tasks efficiently. Methodologies are utilized in various fields, such as software development, project management, business, and education, and can be tailored to meet the specific needs of an organization or project. The methodology for a smart robot for visual impaired individuals is divided into two parts - hardware and software - with the aim of addressing their mobility, communication, education, and safety challenges.

1. Hardware

1) Raspberry pi 4(8 GB).
2) Raspberry Came V2 (8MP).
3) Aluminum Metal Case with Dual Fans for Raspberry Pi.
4) Touch screen display.
5) DC Booster Converter
6) Light sensor.
7) Gas Sensor.
8) Smoke Sensor.
9) Ultrasonic.
10) L298N Motor Driver.
11) Motors.
12) LED 10W.
13) Buzzer 5V (TMB12A05).
14) Speaker 5W.
15) Heatsink.
16) Power supply 5V & 3A.
17) Battery Li-ion Batteries 18650.
18) Lithium Battery Plate.
19) Switch.
20) Water sensor.

2. Software

1) Python.
2) Dart.
3) AI, ML, DL.
4) Object detection.
5) OCR.
6) Text to speech & Speech-to-text.
7) Voice assistance.
8) Face recognition.
9) Linux software.
10) Raspberry OS.
11) Tensorflow.
12) Flutter.
13) Android studio.
14) YOLO.
15) RTOS.
16) Blender 3D.

Design Implementation:

This section describes the process of connecting the various components of the circuit. Initially, the design and placement of each component in the circuit were determined, as depicted in Fig 1. Next, the circuit was connected without regard to the robot's design or structure to verify its proper functioning, as shown in Fig 2. Finally, a comprehensive circuit diagram was created, illustrating all the components and their interconnections, as depicted in Fig 3. This rigorous approach ensured that the circuit was properly designed and connected, laying the foundation for subsequent stages of the project.

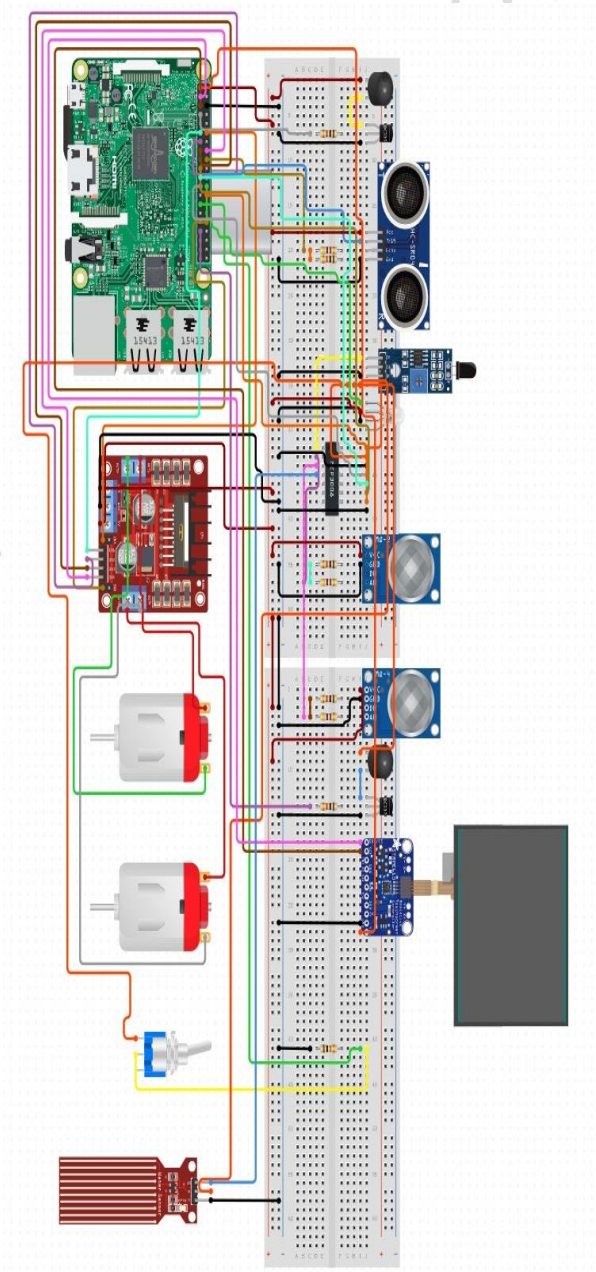


Fig.1

Problem solving:

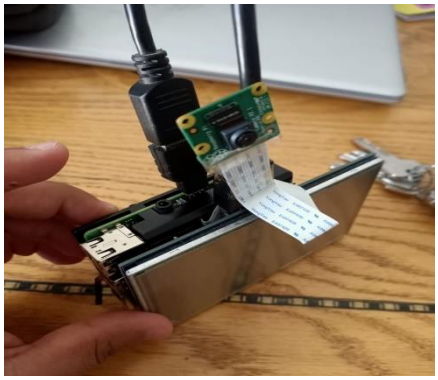


Fig. 2

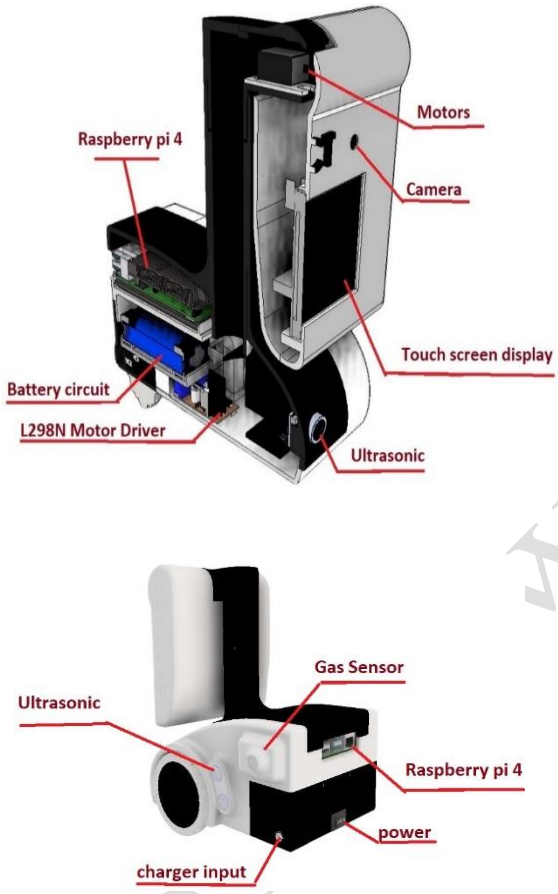


Fig (3)

Raspberry system:

The operating system utilized for the project was Ubuntu Desktop 22.04.2 LTS, which is a long-term support release of the Ubuntu operating system. This version was specifically chosen due to its stability and extended support period of up to five years. It is designed for use with desktop or laptop computers and features the Ubuntu desktop environment alongside pre-installed software packages for everyday activities such as web browsing, email, productivity tools, media playback, and more.

Canonical, the company behind Ubuntu, will provide regular maintenance and security updates for this LTS release throughout its support period, ensuring that users can rely on a stable and secure operating system for their desktop computing needs. The choice of Ubuntu Desktop 22.04.2 LTS serves as a reliable foundation for the project's development and implementation.

Object detection

The proposed design incorporates object detection technology in the robot to enable it to identify objects, people, or animals in its vicinity. This feature ensures a safe environment for users and provides assistance to individuals with disabilities or those performing daily tasks at home.

To implement object detection, the design requires several components. Firstly, YOLO version 8 is used to collect data for identifying objects. Once the training is complete, the resulting training file is utilized in a Python file to obtain the desired output.

The Python code used for this purpose must be adequately prepared. The initial step involves defining the relevant libraries utilized in the code. This ensures that the code can access the necessary tools required for proper execution.

```
import cv2                          # import OpenCV library for computer vision tasks

import argparse                     # import argparse to parse command-line arguments

import pyttsx3                      # import pyttsx3 library for text-to-speech conversion

from ultralytics import YOLO        # import YOLOv8 object detection model from Ultralytics

import supervision as sv            # import supervision module for detection zone and annotation

import numpy as np
```

Once the offices are defined, it is essential to determine the type of version used in the training process. In this case, our proposed design utilizes YOLOv8 for this purpose. YOLOv8 is an advanced version of the You Only Look Once (YOLO) algorithm that utilizes deep learning techniques for object detection and recognition. This version offers improvements in accuracy and speed, enabling quicker and more precise

identification of objects in real-time. By utilizing YOLOv8, the proposed design ensures a reliable and efficient method for detecting objects in the robot's environment.

```
# Define a function to parse command-line arguments

def parse_arguments() -> argparse.Namespace:

    parser = argparse.ArgumentParser(description="YOLOv8 live")

    parser.add_argument ("--webcam-resolution", default=[1280, 720], nargs=2, type=int)

    args = parser.parse_args()

    return args
```

Upon establishment of the object detection and recognition system, it is recommended to include a voice code in the design that utilizes text-to-speech technology to convert written text into audible speech. This feature is expected to improve the user experience for individuals with visual impairments by providing a more natural and intuitive mode of interaction with the robot. The voice code is engineered to convert written commands or responses into synthesized speech output, which enables users to communicate with the robot via voice commands. Integration of this feature is projected to yield a more inclusive and accessible solution for individuals with disabilities, particularly those who are blind and require a

```
# Define a function to convert text to speech

def speak(text):

    engine = pyttsx3.init()

    engine.say(text)

    engine.runAndWait()
```

hands-free means of interacting with the robot.

The main function:

```
# Define the main function

def main():

    args = parse_arguments()

    frame_width, frame_height = args.webcam_resolution

    cap = cv2.VideoCapture(0)

    cap.set(cv2.CAP_PROP_FRAME_WIDTH, frame_width)


cap.set(cv2.CAP_PROP_FRAME_HEIGHT, frame_height)

    model = YOLO("yolov8s.pt")
```

To add a specific box for the person and how to call the person.

```
    # Creating box annotator object to annotate detections with boxes

    box_annotator = sv.BoxAnnotator(thickness = 2, text_thickness = 2, text_scale = 1)

    # Initializing variables for counting detections    prev_labels = []

    count = {}

    speak_interval = 30

    frame_count = 0

    while True:

        ret, frame = cap.read()

        result = model(frame, agnostic_nms=True)[0]

        detections = sv.Detections.from_yolov8(result)

        curr_labels = []

        for _, confidence, class_id, _ in detections:


curr_labels.append(f"{model.model.names[class_id]} {confidence:.2f}")
```

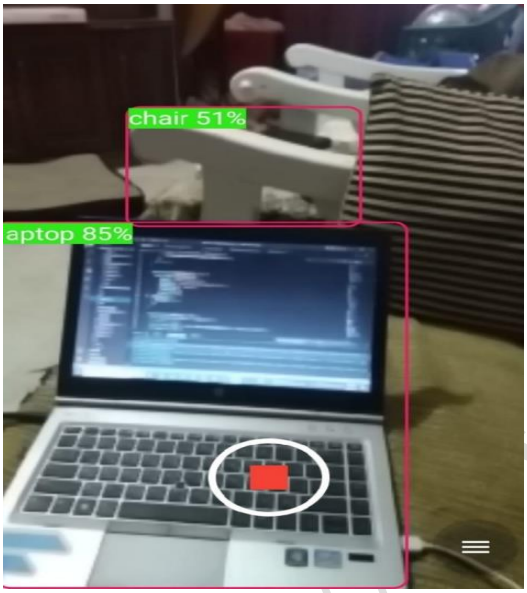


Fig.4

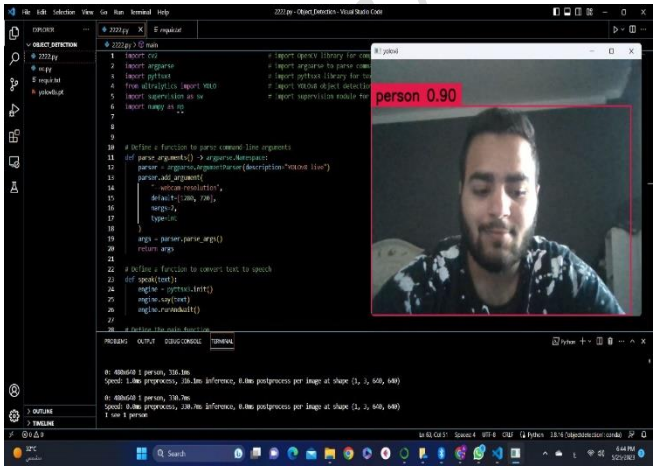


Fig.5

The output of the implemented code, which is illustrated in Fig. 4 and Fig.5. Once the model was trained and some test data was entered, it accurately recognized them. Additionally, experiments were conducted using a mobile phone to identify nearby devices, as shown in Fig.5 Furthermore, the laptop camera was utilized to identify people and their boxes with a high level of accuracy, as depicted in Fig.5.

Name of the device or server used	Google Colab (free)	Laptop (HP 15 da-1882-ne)	ASUS TUF GAMINIG A15	Dell G15 5511 gaming
Data size	4 GB	4GB	4GB	4GB
CPU	Intel Xeon CPU @2.20 GHz	I7 8565u	AMD Ryzen 4600H	I5 11260H
Number of cores	1	4 core – 8 threads	6 Cores - 12 Threads	6 Cores -12 threads
RAM	13GB	16GB	8GB	16GB
GPU Name	Nvidia Tesla T4	Nvidia MX 130	Nvidia GTX1650	Nvidia RTX 3050 ti
Dedicated GPU memory	14.7 GB	2GB	4GB	4GB
GPU Memory	16GB	4GB	4GB	4GB
Time	37.5 hour for (50 epoch)	1100 Hour for (50 epoch) 46 day for (50 epoch)	550 hour for (50 epochs) 23 day for (50 epochs)	21 hour for (50 epochs)

Table1

The Tables 1 and 2 summarize the equipment used in the training process in addition to the time required to train a deep learning model depends on various factors such as the size of the dataset, the complexity of the model architecture, and the hardware used for training. The following tables summarize the time period required to train our computer vision and natural language processing models:

The variation in the training time of the model is due to the graphics card, and to get the best performance, The graphics card that supports CUDA is used to reduce the training time.

The following table shows graphics cards that support CUDA:

GeForce and TITAN Products

GPU	Compute Capability
GeForce RTX 4090	8.9
GeForce RTX 4080	8.9
GeForce RTX 4070 Ti	8.9
GeForce RTX 3090 Ti	8.6
GeForce RTX 3090	8.6
GeForce RTX 3080 Ti	8.6
GeForce RTX 3080	8.6
GeForce RTX 3070 Ti	8.6
GeForce RTX 3070	8.6
Geforce RTX 3060 Ti	8.6
Geforce RTX 3060	8.6
GeForce GTX 1650 Ti	7.5
NVIDIA TITAN RTX	7.5
Geforce RTX 2080 Ti	7.5

GeForce Notebook Products

GPU	Compute Capability
GeForce RTX 4090	8.9
GeForce RTX 4080	8.9
GeForce RTX 4070	8.9
GeForce RTX 4060	8.9
GeForce RTX 4050	8.9
GeForce RTX 3080 Ti	8.6
GeForce RTX 3080	8.6
GeForce RTX 3070 Ti	8.6
GeForce RTX 3070	8.6
GeForce RTX 3060 Ti	8.6
GeForce RTX 3060	8.6
GeForce RTX 3050 Ti	8.6
GeForce RTX 3050	8.6
Geforce RTX 2080	7.5

Table2

Voice assistance:

A Text-to-Speech and Speech-to-Text tool will be utilized as the intermediary between the robot and visually impaired user in our system. Initially, the required libraries for this purpose will be established as in table .:

The main functions that can be performed by the smart robot will be discussed below.

The initial function is designed to assist the visually impaired by announcing the current time and date.

```
# Define a function to tell the current time
def tell_time():
    now = datetime.datetime.now()
    time = now.strftime("%I:%M %p")
    print(f"The current time is {time}")
    speak(f"The current time is {time}")
# Define a function to tell the current date
def tell_date():
    now = datetime.datetime.now()
    date = now.strftime("%A, %B, %d, %Y")
    print(f"Today is {date}")
    speak(f"Today is {date}")
```

The upcoming function is designed to aid the visually impaired by searching for information on Wikipedia and vocalizing the findings.

```
def search_wikipedia():
    print("What do you want to search on Wikipedia?")
    speak("What do you want to search on Wikipedia?")
    # Listen for the user's query
    with sr.Microphone() as source:
        audio = r.listen(source)
        try:
            query = r.recognize_google(audio)
            print(f"Searching Wikipedia for {query}")
            speak(f"Searching Wikipedia for {query}")
            # Get a summary of the query from Wikipedia
            summary = wikipedia.summary(query, sentences=2)
            print(f"According to Wikipedia, {summary}")
            speak(f"According to Wikipedia, {summary}")
        except:
            # Handle any errors in speech recognition or
Wikipedia search
            print("Sorry, I could not understand or find your
query.")
            speak("Sorry, I could not understand or find your
query.")
```

This function is designed to facilitate web browsing on the internet for individuals who are visually impaired.

```
def open_website():
    # Starts listening to the user's voice input with the
microphone
    with sr.Microphone() as source:
        print("Please say the website name.")
        speak("Please say the website name.")
        audio = r.listen(source)
    try:
        # Recognizes the user's voice input and opens the
website in a web browser
        website_name = r.recognize_google(audio)
        webbrowser.open("https://" + website_name + ".com")
        speak(website_name + " is opening.")
    except Exception as e:
        # If there is an error, the code informs the user that it
cannot open the website
        speak("I can't open it.")
```

This function is designed to initiate WhatsApp and transmit a text message to a recipient.

```
def whatsapp():
    # Asks the user what message they would like to send
    print("What message would you like to send?")
    speak("What message would you like to send?")
    with sr.Microphone() as source:
        audio = r.listen(source)
    message = r.recognize_google(audio)
    print(message)
    # Asks the user for the phone number of the recipient
    print("What is the phone number of the recipient?")
    speak("What  is the phone number of the recipient?")
    with sr.Microphone() as source:
        audio = r.listen(source)
    phone_number = r.recognize_google(audio)
    Whats_number =("+20"+ phone_number)
    print(Whats_number)
    # Sends the WhatsApp message using the pywhatkit
library
    pywhatkit.sendwhatmsg_instantly(Whats_number,
message)
    print("WhatsApp message sent successfully")
    speak("WhatsApp message sent successfully")
```

The following function makes jokes and speaks it to the visually impaired user

```
def joke():
    joke = pyjokes.get_joke()
    print(joke)
    speak(joke + "He he he")
```

This function is designed to provide a reminder for something.

```
def set_reminder():
    # Asks the user what they would like to be reminded about
    with sr.Microphone() as source:
        print("What should I remind you about?")
        speak("What should I remind you about?")
        audio = r.listen(source)
        reminder_text = r.recognize_google(audio)
        print("Ok. I will remind you with :" + reminder_text)
        speak("Ok. I will remind you with :" + reminder_text)
        # Asks the user for the time of the reminder
        print("When should I remind you?")
        speak("When should I remind you?")
        for prompt in ["Set hour: ", "Set minutes: ", "am or pm
"]:
            print(prompt)
            speak(prompt)
            audio = r.listen(source)
            try:
                if "hour" in prompt:
                    alarm_hour = int(r.recognize_google(audio))
                    print(alarm_hour)
                elif "minutes" in prompt:
                    alarm_minutes = int(r.recognize_google(audio))
                    print(alarm_minutes)
                else:
                    am_pm_str =
r.recognize_google(audio).replace("b","p").replace(".",
"").replace("B&M", "pm")
                    print(am_pm_str)
            except ValueError:
                # If there is an error, the speaker informs the user
that it cannot understand and prompts them to try again
                speak("Sorry, I didn't understand. Please try
again.")
                break
    # Formats the time of the reminder based on whether it is
AM or PM
    am_pm = am_pm_str.lower()
    print(f"Waiting for time: {alarm_hour}:{alarm_minutes}
{am_pm}")
    speak(f"Waiting for time: {alarm_hour}:{alarm_minutes}
{am_pm}")
    if am_pm == 'pm' and alarm_hour < 12: alarm_hour += 12
    elif alarm_hour == 12 and am_pm == 'am': alarm_hour -=
12
    # Waits until it is time for the reminder to be sent, then
plays an alarm sound and reminds the user of their reminder
    while True:
        now = datetime.datetime.now()
        if now.hour == alarm_hour and now.minute ==
alarm_minutes:
            playsound.playsound('alarm.mp3')
            speak("\nIt's the time!\n "+ reminder_text)
            break
        time.sleep(1)
```

The following function is responsible for the set a timer with specific time.

```
def set_timer():
    # Print a message asking the user how long he wants to set
the timer for.
    print("How long do you want me to set the timer for?")
    speak("How long do you want me to set the timer for?")
    # Get the user's input for the duration of the timer.
    duration = get_audio()
    # If the user did not provide any input, return.
    if duration == "None":
        return
    # Parse the user's input to extract the hours, minutes, and
seconds
    hours = 0
    minutes = 0
    seconds = 0
    matches = re.findall(r"\d+", duration)
    # If the user provided only one number, assume it is the
number of seconds
```

```
    if len(matches) == 1:
        seconds = int(matches[0])
    # If the user provided two numbers, assume they are minutes and seconds
    elif len(matches) == 2:
        minutes = int(matches[0])
        seconds = int(matches[1])
    # If the user provided three numbers, assume they are hours, minutes, and seconds
    elif len(matches) == 3:
        hours = int(matches[0])
        minutes = int(matches[1])
        seconds = int(matches[2])
    # If the user provided more than three numbers or non-numeric input, inform him that the input was invalid and exit the function
    else:
        print("Sorry, I couldn't parse that duration. Please try again.")
        speak("Sorry, I couldn't parse that duration. Please try again.")
        return
    # Convert the hours, minutes, and seconds to a total number of seconds
    total_seconds = hours * 3600 + minutes * 60 + seconds
    # If the total number of seconds is greater than 0, inform the user of the timer duration and set a timer for that length
    if total_seconds > 0:
        print(f"OK, I will set the timer for {hours} hours, {minutes} minutes, and {seconds} seconds.")
        speak(f"OK, I will set the timer for {hours} hours, {minutes} minutes, and {seconds} seconds.")
        time.sleep(total_seconds)
        # When the timer goes off, inform the user and play an alarm sound
        print(f"Time's up!")
        speak(f"Time's up!")
        # play an alarm sound file
        playsound.playsound('alarm.mp3')
        # If the total number of seconds is not greater than 0, inform the user that the duration was not valid
    else:
        print("Sorry, that's not a valid duration.") speak("Sorry, that's not a valid duration.")
```

- The following function is responsible for taking a note and save it in a text file

```
def note():
    # Start recording audio from the microphone
    with sr.Microphone() as source:
        # speak a message asking the user to say their note.
        print("Say your note")
        speak("Say your note")
        # Record the audio for 10 seconds
        audio = r.record(source, duration=10)
        # Use Google's speech recognition API to transcribe the audio into text
        note = r.recognize_google(audio)
        # Speak the transcribed note to the user and save it to a file called "note.txt"
        print("Your note is: " + note + "and it is saved successfully")
        speak("Your note is: " + note + "and it is saved successfully")
    with open("note.txt", "w") as f:
        f.write(note)
        print("Note saved to your notes")
        speak("Note saved to your notes")
```

The following function is responsible for the voice assistant, which works using artificial intelligence (Ai)

```
def voice_assistant():
    """ * Runs thAe voice assistant.
        * Continually prompts the user for a question, and then asks ChatGPT to answer it. """
    while True:
        # Setting up the OpenAI API key and initializing chat_log to store conversation history
        openai.api_key = "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"
        chat_log = []
        # Initializing speech recognizer instance
        with sr.Microphone() as source:
            # Asking the user for input by generating speech using the speak() function
            print("What do you want to ask ChatGPT about?")
            speak("What do you want to ask ChatGPT about?")
            # Recording the user's input
            audio = r.listen(source)
            # Recognizing the audio input and storing the text in a variable
            Target_question = r.recognize_google(audio)
            # Calling the get_audio() function to get the user's input from microphone
            user_message = get_audio()
            # Generating speech to confirm the user's input
            print("Sure, I will search for \"" + Target_question + "\".")
            speak("Sure, I will search for \"" + Target_question + "\".")
            # If the user inputs "quit", break out of the loop
            if user_message.lower() == "quit":
                break
            else:
                # Storing the user's input in chat_log
                chat_log.append({"role": "user", "content": user_message})
                response = openai.ChatCompletion.create(
                    model = "gpt-3.5-turbo",
                    # Passing the chat history to OpenAI's API for generating a response
                    messages = chat_log
                )
                # Extracting the generated response from the API's response
                assistant_response = response['choices'][0]['message']['content']
                # Printing the generated response
```

```
            print("ChatGPT: ",
assistant_response.strip("\n").strip())
            # Generating speech for the generated response
            speak("ChatGPT: ",
assistant_response.strip("\n").strip())
            # Storing the generated response in chat_log.
            chat_log.append({"role": "assistant", "content":
assistant_response.strip("\n").strip()})
            print(response)
```

The following function is responsible for translating from English into Arabic.

```
def translate():
    translator = googletrans.Translator()
    speak("Say something in English")
    with sr.Microphone() as source:
        audio = r.listen(source)
        text = r.recognize_google(audio)
        print(f"You said: {text}")
        # Set the target language
        target_language = "arabic"  # or extract the target
language from the text using string manipulation
        # Translate the text
        translated_text = translator.translate(text,
dest=target_language).text
        speak(f"Translated to {target_language}:
{translated_text}")
        print(translated_text)
```

The following function is responsible for downloading videos from youtube in specific quality

```
def Download():
    link = input("Put your YouTube link here! URL: ")
    youtube_object = YouTube(link)
    streams = youtube_object.streams.filter(progressive=True)
    print("Available resolutions:")
    speak("Available resolutions:")
    # Loop through each stream and print its resolution and
index
    for i, stream in enumerate(streams):
        print(f"{i}: {stream.resolution}")
        speak(f"{i}: {stream.resolution}")
    # prompt user to select a resolution
    while True:
        print("please say the number corresponding to the
resolution that you want")
        speak("please say the number corresponding to the
resolution that you want")
        selection = get_audio()
        # If the input is not None and is a digit
        if selection is not None and selection.isdigit():
            # Convert the selection to an integer
            selection = int(selection)
            # If the selection is within the range of available
streams, break the loop
            if selection >= 0 and selection < len(streams):
                break
    selected_stream = streams[selection]
    file_path = selected_stream.download()
    print(f"Your video was downloaded to {file_path}.")
    print("This download has completed! Yeeeah!")
    speak("This download has completed! Yeeeah!")
```

- The following function is responsible for giving a full weather report in specific area.

```
def get_weather():
    with sr.Microphone() as source:
        speak("Which location?")
        audio = r.listen(source)
        location = r.recognize_google(audio)
        # Print the location
        print("Location: " + location)
    # Get the weather manager
    mgr = owm.weather_manager()
    # Get the observation for the location
    observation = mgr.weather_at_place(location)
    # Get the weather details from the observation
    w = observation.weather
    # Get temperature, wind speed, and humidity
    temperature = w.temperature('celsius')['temp']
    # Convert wind speed from m/s to km/h
    wind_speed_kmh=str(round((w.wind()['speed']) * 3.6, 2))
    # Get the humidity percentage
    humidity = w.humidity
    # Create a report string with the weather details
    weather_report = f"The temperature in {location} is
{temperature} degrees Celsius. The wind speed is
{wind_speed_kmh} kilometers per hour. The humidity is
{humidity} percent."
    # Print the weather report and speak it
    print(weather_report)
    speak(weather_report)
```

- The following function is used to convert speech to text and save it in a PDF files.

```
def pdf():
    with sr.Microphone() as source:
        print("Say the content of the pdf...")
        speak("Say the content of the pdf...")
        audio = r.listen(source)
    # Convert speech to text
    text = r.recognize_google(audio)
    print(f"You said: {text}")
    speak(f"You said: {text}")
# Define the data for each page
    data = [text]
    # Create a PDF with multiple pages
    pdf_filename = "multi_page.pdf"
    doc = SimpleDocTemplate(pdf_filename, pagesize=letter)
    for page_content in data:
        # Initialize the story for this page
        story = []
        # Split the content into paragraphs based on ".   " line
breaks
        paragraphs = page_content.split(".   ")
        for paragraph in paragraphs:
            # Add the paragraph to the story
            p = Paragraph(paragraph)
```

```
        story.append(p)
        # Add a spacer after each paragraph
        story.append(Spacer(1, 0.2 * inch))
    # Add the story to the document
    doc.build(story)
print("PDF saved successfully")
speak("PDF saved successfully")
```

Face recognition

Facial recognition technology is utilized to identify individuals who have been added to a smart robot designed for the visually impaired, providing a significant advantage in terms of security by enabling the detection of specific users. This feature facilitates the identification of potential theft cases and promptly alerts relevant authorities. The integration of facial recognition technology in smart robots for the blind marks a notable advancement in security and has the potential to significantly enhance the protection of valuable resources.
The libraries used:

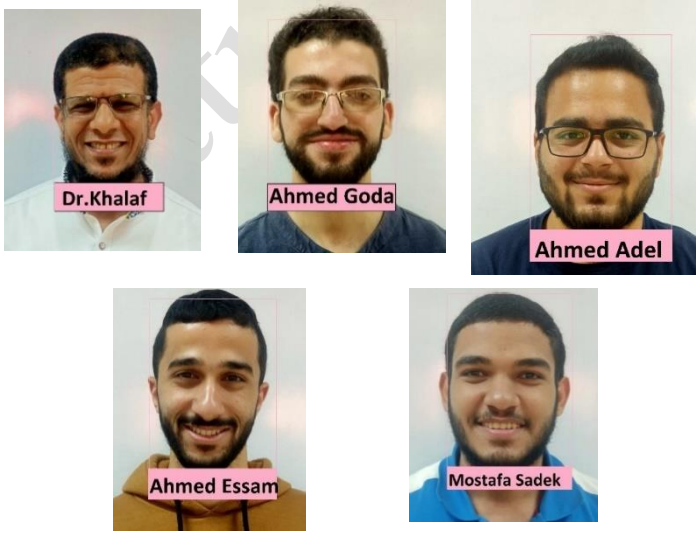
```
import face_recognition   # provides facial recognition capabilities through ML algorithms.
import cv2                # used for real-time image and video processing, object detection, and other computer vision tasks.
import numpy as np        # a library for scientific computing in Python that provides support for multi-dimensional arrays and mathematical operations on them
import pyttsx3            # a library for text-to-speech conversion, which means it can convert written text into spoken words
import os                 # a built-in Python library for interacting with the operating system, which can be used to access files and directories, manipulate environment variables, and more
```

The complete code:

```
engine = pyttsx3.init()
# Get a reference to webcam #0 (the default one)
video_capture = cv2.VideoCapture(0)
filenames = []
with open("filenames.txt", "r") as file:
# Read the array from the text file, using the `numpy.loadtxt()` function.
    known_face_names = np.loadtxt(file, delimiter=",", dtype=str)
with open("all.txt", "r") as file:
# Read the array from the text file, using the `numpy.loadtxt()` function.
    content = np.loadtxt(file, delimiter=",")
    known_face_encodings = content
# Initializing variables to store face locations, encodings and names for each frame.
face_locations = []
face_encodings = []
face_names = []
frame_count = 0
prev_num_faces = 0
while True:
    # Grab a single frame of video
    ret, frame = video_capture.read()
    if frame_count % 25 == 0:  # check for matches and speak the name every 30 frames
        # Resize frame of video to 1/5 size for faster face recognition processing
        small_frame = cv2.resize(frame, (0, 0), fx=0.2, fy=0.2)
        # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
        rgb_small_frame = small_frame[:, :, ::-1]
        # Find all the faces and face encodings in the current frame of video
        # Number_of_times_to_upsample parameter specifies how many times to upsample the image.
        face_locations = face_recognition.face_locations(rgb_small_frame, number_of_times_to_upsample=3)
        face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations, num_jitters=5)
        # Initialize an empty list for storing the names of recognized faces
        face_names = []
        # Comparing the face encodings of the reduced frame with the known face encodings
        for face_encoding in face_encodings:
            # See if the face is a match for the known face(s)
            matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
            name = "Unknown"
            # Calculating the face distances between the encodings to identify the best match.
            face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
            # Find the index of the closest match.
            best_match_index = np.argmin(face_distances)
            # Checks if the closest match is below the tolerance level.
            if matches[best_match_index]:
                # If it is, then the name associated with that known_face_encoding is assigned to the detected face.
                name = known_face_names[best_match_index]
            # Adds the assigned name to the list of face_names.
            face_names.append(name)
        # Counts the number of faces detected in the current frame.
        num_faces = len(face_locations)
        # checks if the number of detected faces has changed since the last frame.
        if num_faces != prev_num_faces:
            names_str = ", ".join(face_names)
            # If it has, then it speaks out the number of persons detected along with their names.
            engine.say(f"{num_faces} persons detected: {names_str}")
            engine.runAndWait()
```

```
        # updates the prev_num_faces variable with the current number of detected faces so that it can be used in the next frame.
        prev_num_faces = num_faces
    frame_count += 1
    # iterate through each face location and name returned by the model
    for (top, right, bottom, left), name in zip(face_locations, face_names):
        # scale up the coordinates of the face locations to match the original frame size
        top *= 0
        right *= 5
        bottom *= 5
        left *= 4
        # Draw a box around the face with White color and thickness of 2
        cv2.rectangle(frame, (left, top), (right, bottom), (255, 230, 249), 2)
        # Draw a label with a name below the face with Black color
        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (255, 230, 249), cv2.FILLED)
        # specify font type and size for the label text
        font = cv2.FONT_HERSHEY_DUPLEX
        # add person's name as text to the labeled rectangle with white color and thickness of 1
        cv2.putText(frame, name, (left + 5, bottom - 6), font, 1.25, (0, 0, 0), 1)
    # Display the resulting image
    cv2.imshow('Video', frame)
    # Hit 'q' on the keyboard to quit!
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
video_capture.release()
cv2.destroyAllWindows()
```

The following images are the output of the Face recognition code:
This part displays the output of the code as it appears in Fig .6



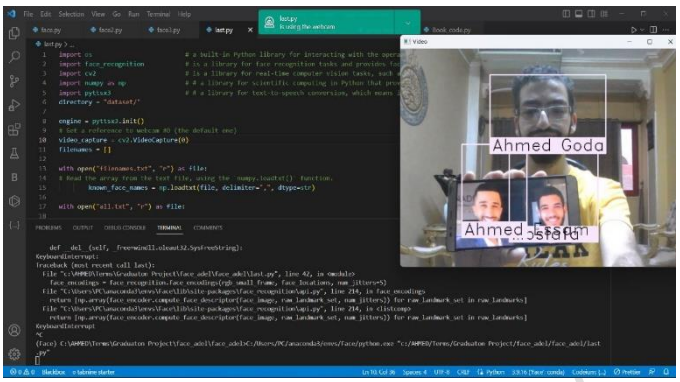
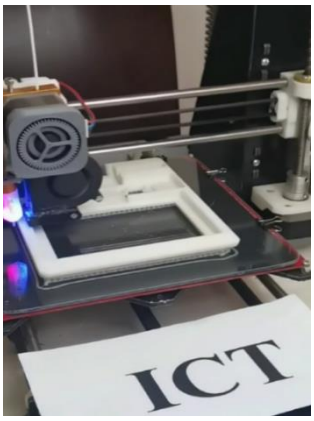


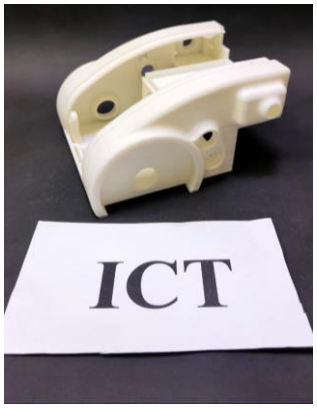
Fig.6.

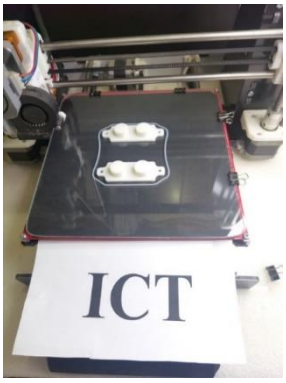
3D Design

- Blender is a powerful and versatile 3D modeling software that can be used for a wide range of projects. Here are some of the reasons why you might want to use Blender:
    1. **It's free:** Blender is open-source software, which means it's free to download and use.
    2. **Cross-platform:** Blender works on Windows, Mac, and Linux, so you can use it regardless of your operating system.
    3. **3D modeling:** Blender has robust tools for creating 3D models, including sculpting, texturing, rigging, and animation.
    4. **Simulation:** Blender has features for simulating things like fluid dynamics, cloth, and particle systems, making it a great tool for visual effects work.
    5. **Python scripting:** Blender has an extensive Python API, which allows you to automate tasks and customize the software to suit your needs.
- Overall, Blender is a great choice for anyone looking to get into 3D modeling, animation, or game development.
- Printing period, number of pieces, materials used, and type of printing:
    - Number of hours: approximately 180 hours
    - Number of printed pieces: 23 pieces
    - Raw materials: PLA+ & TPU
    - The automatic printer: prusa i3 mk3s
    - Print Type: Fused Deposition Modelin (FDM)

Below are pictures from within the program that show the shape of the design:
In this part, we clarify the shape of the design as in Fig .7. and Fig.8, this is the design shown on the Blender program, and after that there the images in Fig 9. pictures showing the stages of printing the design in 3D printing, and in Fig .10 a picture after the completion of printing and installing the design without components.







Conclusion

The development of a smart robot for blind people is a significant step forward in creating an integrated and independent living experience. With the increasing challenge of navigating around unfamiliar spaces, it is essential to have a solution that can address the challenges of mobility and orientation faced by the visually impaired. The smart robot application provides a comprehensive solution by integrating and controlling multiple smart devices in a seamless and intuitive manner, to help blind people navigate around their homes and surroundings.

The smart robot application will improve the overall efficiency and convenience of living for the visually impaired by coordinating various devices to create desired environments, such as turning on the lights, adjusting the temperature, and setting the security system when the homeowner arrives home. The application will also optimize the performance of the various devices to ensure that they are working together optimally and providing maximum assistance to the visually impaired.

Additionally, the smart robot for blind people will be equipped with advanced sensors and navigation systems to help them move around the house without any assistance. It will use machine learning algorithms to understand the layout of the house and the location of different devices, to provide accurate directions to the user and prevent collisions. Through voice commands or touch screen, the robot will interact with the user to make sure they are aware of their surroundings and providing valuable assistance where necessary.

Overall, the smart robot for blind people will enhance the quality of life and independence of visually impaired individuals, making it easier for them to navigate and perform daily tasks with ease and greater confidence.

Furthermore, the smart robot application will provide a simple and user-friendly interface that allows homeowners to control all their smart devices from a single, unified view. The application will also be compatible with a variety of smart devices and platforms, making it easier for homeowners to adopt and integrate it into their existing smart home setup.

Furthermore, by offering a platform that homeowners, device makers, and service providers can quickly embrace, the smart robot application has the potential to revolutionize the smart home sector. As a result, the market for smart homes may experience increased innovation and growth, opening up new chances for firms and entrepreneurs to introduce innovative goods and services.

Additionally, the smart robot application has the potential to make smart homes more accessible to a wider range of consumers, including those with disabilities, elderly individuals, and others who may face challenges in controlling and managing their smart devices. The application's user-friendly interface and voice control capabilities can make it

easier for these individuals to control and monitor their smart home, improving their quality of life and independence.

The smart robot application for a smart home will play a critical role in creating an efficient and convenient living experience for homeowners. With its integrated and user-friendly approach, the smart robot application is poised to become the next generation of smart home solutions, providing homeowners with the peace of mind and comfort they deserve.

We worked on Python version 3.9.16 for several reasons. The first reason is that this version is the most stable version. Secondly, all the libraries used in the codes are compatible with it.

We used the Dell G15 5511 gaming to produce a training file by training 120,000 images over 21 hours using the device specifications: CPU I5 11260H, GPU RTX 3050 ti, RAM 16GB.

Creating the smart robot 3D print design, took 27 days, and to print 3D printing, it took 10 days.

In conclusion, the smart robot application for a smart home is not only a solution to the current challenges faced by homeowners, but it also represents a new opportunity for businesses and entrepreneurs to innovate and create new products and services. With its potential to revolutionize the smart home industry and make smart homes more accessible to a wider range of consumers, the smart robot application has a bright future ahead.

References

[1] International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Published by, www.ijert.org ICACT - 2016 Conference Proceedings Special Issue - 2016.

[2] Ultrasonic Blind Stick for Completely Blind People to Avoid Any Kind of Obstacles Arnesh Sen Kaustav Sen Jayoti Das Jadavpur University: Dept. of Physics, Kolkata, India.

[3] PAPER • OPEN ACCESS Raspberry PI Based Smart Walking Stick to cite this article: M. Ganesan et al 2020 IOP Conf. Ser.: Mater. Sci. Eng. 981 042090.

[4] Yoann Dieudonné, Shlomi Dolev, Franck Petit, Michael Segal. Deaf, Dumb, and Chatting Robots, Enabling Distributed Computation and Fault-Tolerance Among Stigmergic Robots. [Research Report] 2009, pp.15.

[5] International Journal of Computer Trends and Technology Volume 68 Issue 3, 13 15, March 2020 ISSN: 2231-2803 / https://doi.org/10.14445/22312803/IJCTT-V68I3P103 2020 Seventh Sense Research Group.

[6] Volume 5, Issue 4, April – 2020 International Journal of Innovative Science and Research Technology ISSN No:-2456-2165.

[7] Benachir B. Nouhaila ,Adekunle A. Adepoju(2022),Application of Aloe Vera-derived Plant-based Cell in Powering Wireless IoT devices in a Smart Greenhouse. IUSRJ International Uni-Scientific Research Journal (3)(19),126-132. https://doi.org/10.59271/s44768.022.2010.19

[8] Raspberry pi 4: https://static.raspberrypi.org/files/product-briefs/200521+Raspberry+Pi+4+Product+Brief.pdf.

[9] Raspberry Pi Camera v2: https://www.farnell.com/datasheets/2056179.pdf?_ga=1.152577328.880870297.1479740269

[10] Touch screen display: https://www.openhacks.com/uploadsproductos/hdmi_interface_5_inch_800x480_tft_display_-_elecrow.pdf.

[11] Power Supply: https://static.raspberrypi.org/files/product-briefs/USB-C-Product-Brief.pdf.

[12] PHOTOSENSITIVE RESISTANCE SENSOR: https://hobbycomponents.com/sensors/832-photosensitive-resistance-sensor-module.

[13] Water sensor: https://curtocircuito.com.br/datasheet/sensor/nivel_de_agua_analogico.pdf.

[14] Motors: https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/3777_Web.pdf.

[15] Motor Driver L298N: http://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf.

[16] Speaker: https://www.electroncomponents.com/4-inch-speaker-4-ohm-5watts.

[17] Ultrasonic: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf.

[18] Buzzer: http://electroniccomponentsindia.blogspot.com/2014/09/buzzers.html.

[19] Esp32: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[20] GPU: https://developer.nvidia.com/cuda-gpus

[21] Flutter

- https://www.javatpoint.com/flutter
- https://www.fullstacklabs.co/blog/introduction-to-flutter
- https://www.tutorialspoint.com/flutter/flutter_introduction.htm
- https://litslink.com/blog/why-should-you-build-your-next-app-with flutter
- https://www.zrix.com/blog/flutter-mobile-application-development
- https://digitalskynet.com/blog/Ionic-vs-Flutter-vs-React-Native

[22] Python:

- https://www.javatpoint.com/python-tutorial
- https://techvidvan.com/tutorials/python-advantages-and-disadvantages/

[23] Python GUI:

- https://www.gartner.com/en/information-technology/glossary/gui-graphical-user-interface
- https://www.activestate.com/blog/top-10-python-gui-frameworks-compared/

[24] OCR:

- https://aws.amazon.com/what-is/ocr/?nc1=h_ls
- https://www.docacquire.com/resources/blog/what-is-ocr/
- https://www.geeksforgeeks.org/advantages-and-disadvantages-of-optical-character-reader-ocr/
- https://www.docdigitizer.com/blog/what-is-ocr/

[25] Raspberry Pi OS:

- https://www.tutorialspoint.com/raspberry_pi/raspberry_pi_quick_guide.htm
- https://picockpit.com/raspberry-pi/raspberry-pi-os-overview/
- https://linuxhint.com/what-is-raspberry-pi/
- https://robu.in/5-pros-and-5-cons-of-raspberry-pi/
- https://raspberrytips.com/raspberry-pi-pros-and-cons/
- https://www.educba.com/uses-of-raspberry-pi/
- https://www.oreilly.com/library/view/raspberry-pi-amazing/9781787128491/ch24s06.html

[26] Speech-to-text:

- https://aws.amazon.com/what-is/speech-to-text/
- https://www.amberscript.com/en/blog/how-speech-to-text-software-works/#one
- https://itchronicles.com/speech-to-text/
- https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Text-to-Speech-Conversion.html
- https://studylib.net/doc/7569205/advantages-and-disadvantages-of-voice-recognition---pac-itgs
- https://itchronicles.com/speech-to-text/
- https://leaddesk.com/blog/speech-to-text-guide-for-contact-centers/.

[27] Voice Assistance:

- https://www.santander.com/en/stories/everything-you-need-to-know-about-voice-assistants
- https://alan.app/blog/voiceassistant-2/
- https://www.zdnet.com/home-and-office/smart-home/12-smart-home-devices-that-make-great-gifts-in-2022/
- https://www.miquido.com/blog/what-are-voice-assistants/
- https://blogs.oracle.com/marketingcloud/post/advantages-and-disadvantages-of-voice-assistants-for-marketers
- https://www.techwalla.com/articles/the-disadvantages-of-voice-recognition-software
- https://www.techulator.com/resources/18784-the-12-advantages-and-disadvantages-of-voice-user-interface
- https://www.workstatus.io/blog/the-pros-and-cons-of-voice-assistants-in-the-workplace/

[28] Flask:

- https://www.tutorialspoint.com/flask/flask_tutorial.pdf
- https://www.analyticsvidhya.com/blog/2021/10/easy-introduction-to-flask-framework-for-beginners/

[29] AI, ML, and DL:

- http://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html
- https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f
- https://codebots.com/artificial-intelligence/the-3-types-of-ai-is-the-third-even-possible
- https://www.datasciencecentral.com/ai-ml-or-dl-learn-what-it-means\
- https://hypersense.subex.com/blog/ai-vs-ml-vs-dl-whats-the-difference/

[30] object detection:

- https://d2l.ai/chapter_computer-vision/bounding-box.html#bounding-box
- https://lilianweng.github.io/lil-log/2017/10/29/object-recognition-for-dummies-part-1.html
- 3-https://www.v7labs.com/blog/object-detection-guide#h1
- https://medium.com/ml-research-lab/what-is-object-detection-51f9d872ece7 is

[31] TensorFlow:

- https://www.guru99.com/what-is-tensorflow.html#9
- https://www.javatpoint.com/advantage-and-disadvantage-of-tensorflow
- https://www.techtarget.com/searchdatamanagement/definition/TensorFlow
- https://www.javatpoint.com/tensorflow

[32] Ubuntu on a Raspberry Pi:

- https://ubuntu.com/download/raspberry-pi