UNIVERISTY OF PIRAEUS - DEPARTMENT OF INFORMATICS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

MSc «Advanced Informatics and Computing Systems - Software Development and Artificial Intelligence»

ΠΜΣ «Προηγμένα Συστήματα Πληροφορικής - Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

MSc Thesis

Μεταπτυχιακή Διατριβή

Thesis Title: Τίτλος Διατριβής:	**AISA : AI Smart Home Assistant, development of hardware and software for a home assistant intelligent robot** AISA : AI Smart Home Assistant, ανάπτυξη Υλικού και Λογισμικού ευφυούς ρομπότ οικιακής χρήσης
Student's name-surname: Ονοματεπώνυμο φοιτητή:	**Borsis Panagiotis** Μπόρσης Παναγιώτης
Father's name: Πατρώνυμο:	**Dimitrios** Δημήτριος
Student's ID No: Αριθμός Μητρώου:	ΜΠΣΠ19034
Supervisor: Επιβλέπων:	**Themistoklis Panagiotopoulos, Professor** Θεμιστοκλής Παναγιωτόπουλος, Καθηγητής

October 2023/ Οκτώβριος 2023

3-Member Examination Committee

Τριμελής Εξεταστική Επιτροπή

Themistoklis Panagiotopoulos Professor	**Aggelos Pikrakis Assistant Professor**	**Ioannis Tasoulas Assistant Professor**
Θεμιστοκλής Παναγιωτόπουλος Καθηγητής	Άγγελος Πικράκης Επίκουρος Καθηγητής	Ιωάννης Τασούλας Επίκουρος Καθηγητής

1 INDEX

2 ΠΕΡΙΛΗΨΗ

Σήμερα η τεχνητή νοημοσύνη έχει μπει στις ζωές μας όσο ποτέ άλλοτε. Συνεχώς βλέπουμε νέα επιτεύγματα, που ανοίγουν δρόμους για ακόμα μεγαλύτερα στο μέλλον. Ένας τομέας που επηρεάζεται άμεσα από την ραγδαία ανάπτυξη της τεχνητής νοημοσύνης είναι η ρομποτική.

Εκατομμύρια ρομπότ, διαφόρων μεγεθών και λειτουργιών, έχουν κατασκευαστεί ώστε να μπορέσουν να βελτιώσουν την καθημερινότητα των ανθρώπων. Πόσο προσιτά μπορεί να είναι όμως αυτά στον μέσο καταναλωτή; Τι μπορεί να γίνει για να περιοριστεί το κόστος κατασκευής τους;

Με αυτές τις ανησυχίες κατά νου αποφάσισα ως αντικείμενο της μεταπτυχιακής διατριβής μου, την κατασκευή ενός έξυπνου οικιακού βοηθού ρομπότ. Χρησιμοποίησα όσο το δυνατόν φθηνότερα υλικά και ξεκίνησα ένα ταξίδι το οποίο με συνεπήρε. Έδωσα τον καλύτερο μου εαυτό σε αυτό το ρομπότ και θεωρώ ότι τα περιθώρια βελτίωσης του είναι τεράστια ως προς το λογισμικό. Διαθέτοντας 3d printer και ένα raspberry pi, καθώς και μερικούς αισθητήρες χαμηλού κόστους, έφτιαξα μια συσκευή που μπορεί να ανταποκριθεί πολύ καλά στις απαιτήσεις που μπορεί να έχει κάποιος από ένα οικιακό βοηθό, με την αλληλεπίδραση που μπορεί να δώσει ένα ρομπότ σε μία εφαρμογή. Παρακάτω θα αναλύσω το πως κατασκευάστηκε τόσο το ρομπότ όσο και ο κώδικας του.

3 ABSTRACT

Nowadays, artificial intelligence has entered our lives like never before. We are constantly seeing new achievements, which pave the way for even greater ones in the future. One area directly affected by the rapid development of artificial intelligence is robotics.

Millions of robots, of various sizes and functions, have been built to improve people's daily lives. But how accessible can these be to the average consumer? What can be done to limit their construction cost as low as possible?

Keeping that in mind I decided as the subject of my master's thesis, the construction of an intelligent robotic home assistant. I used the cheapest possible materials and started a journey which captivated me. I gave this robot my best shot and I think it has a lot of room for improvement in terms of software. Having a 3d printer and a raspberry pi, as well as some low-cost sensors, I made a device that can very well meet the demands of a home helper, with the interactivity that a robot can give to an application. Below I will analyze how both the robot and its code were built.

4 INTRODUCTION

In recent years, the fusion of robotics and artificial intelligence has led to remarkable advancements in various domains. From industrial automation to healthcare assistance, robots have transformed the way tasks are accomplished, exhibiting efficiency, accuracy, and adaptability. This thesis delves into the exploration and development of an intelligent robotic system that encompasses cutting-edge technologies, including computer vision, machine learning, and natural language processing, to create an interactive and versatile companion.

The primary objective of this thesis is to design and implement a robot with advanced perception and recognition capabilities, always keeping in mind to keep the cost low so that it is accessible from anyone. Being based on a Raspberry pi and a Google Coral USB accelerator the robot excels in object recognition, allowing it to identify and classify various objects in its environment. Furthermore, it possesses person recognition capabilities, enabling it to detect and differentiate individuals, forming the basis for personalized interactions and tailored assistance. Leveraging the power of reinforcement learning, the robot can continuously improve its person recognition abilities, adapting to a diverse range of individuals.

In addition to its recognition capabilities, the robot incorporates gesture recognition, enabling it to interpret and respond to human gestures, thereby fostering intuitive communication. It further demonstrates the ability to locate and track movement, ensuring its awareness of dynamic changes in the environment. An innovative aspect of this research is the integration of emotion detection, empowering the robot to perceive and respond to human emotions, thereby establishing a more empathetic and engaging interaction.

Furthermore, the robot possesses the ability to perform object and person counting within its field of view, providing valuable insights for applications such as crowd monitoring and occupancy analysis. With the integration of motor control, the robot can execute various movement actions, enhancing its physical interaction and mobility.

This thesis also explores an intriguing dimension of the robot's capabilities—the ability to detect signs of drowsiness in users. By monitoring specific cues and employing machine learning algorithms, the robot can determine if a user is displaying signs of sleepiness, leading to potential applications in alertness management and safety.

Moreover, the robot utilizes three distinct chatbots, namely ChatGPT, Google Assistant, and Chatsonic, to facilitate seamless communication and information provision to users. These chatbots offer a diverse range of conversational capabilities, enabling the robot to engage in natural language dialogue, answer queries, and provide relevant information, thereby enhancing user interaction and satisfaction.

Through comprehensive experimentation and evaluation, this thesis aims to demonstrate the effectiveness and limitations of the developed intelligent robotic system. Real-world scenarios and performance metrics will be presented to showcase the robot's capabilities and its potential impact in various domains. The findings of this research contribute to the advancement of human-robot interaction, inspiring future innovations and research endeavors in the field of intelligent robotic systems.

Finally, a full guide to build the robot is provided, along with code explanations and a head-to-head analysis of the three chatbots that are used. In addition, there is also a useful comparison between the usage of Coral USB Accelerator or not on low end devices like a Raspberry pi.

In conclusion, this thesis presents a comprehensive exploration of an intelligent robotic system that combines computer vision, machine learning, and natural language processing to facilitate enhanced human-robot interaction. The robot's capabilities in object and person recognition, gesture interpretation, movement tracking, emotion detection, counting, motor control, drowsiness detection, and conversational engagement make it a valuable and versatile companion. The outcomes of this research have the potential to drive significant progress in robotics and contribute to the realization of intelligent robotic systems in diverse applications.

Please note that this prologue can be further customized and expanded based on your specific contributions, research methodology, and objectives.

5 CREATING THE ROBOT

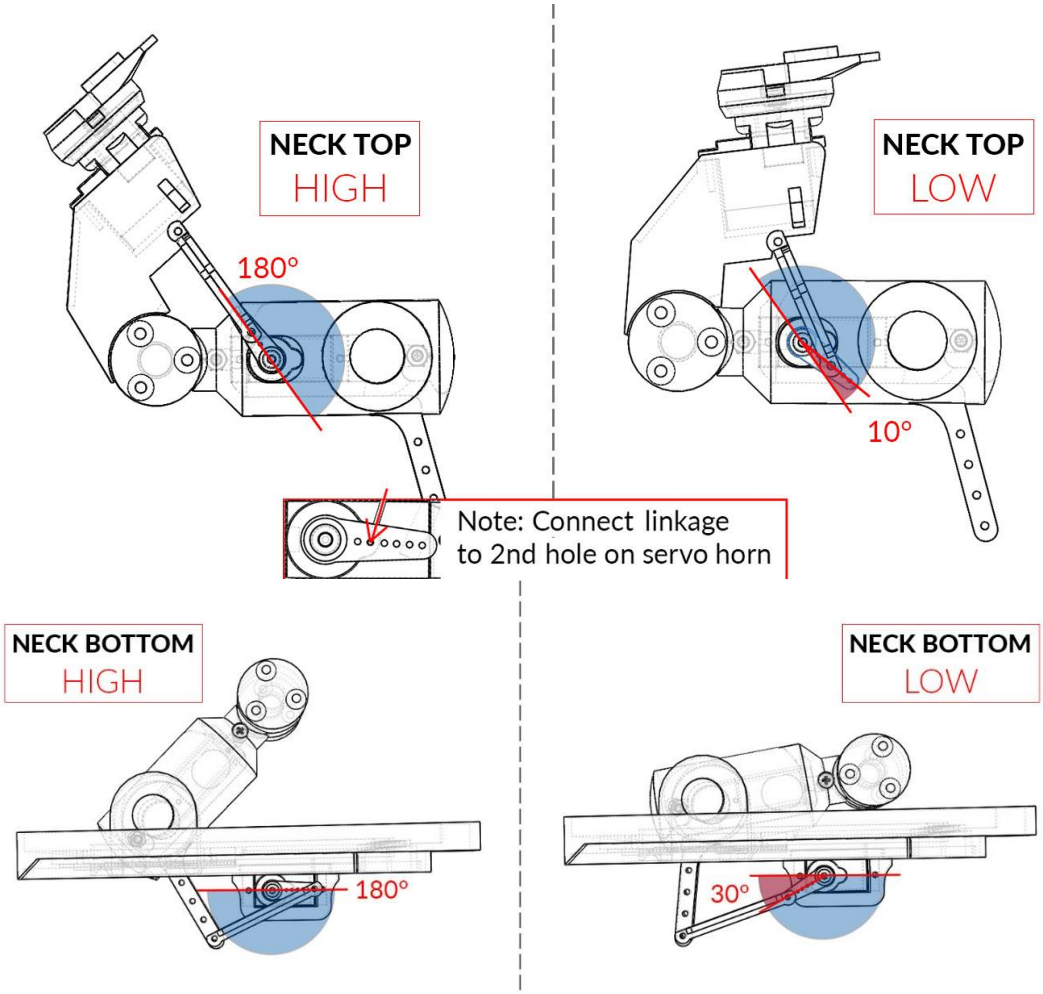
5.1 COLLECTING THE PARTS

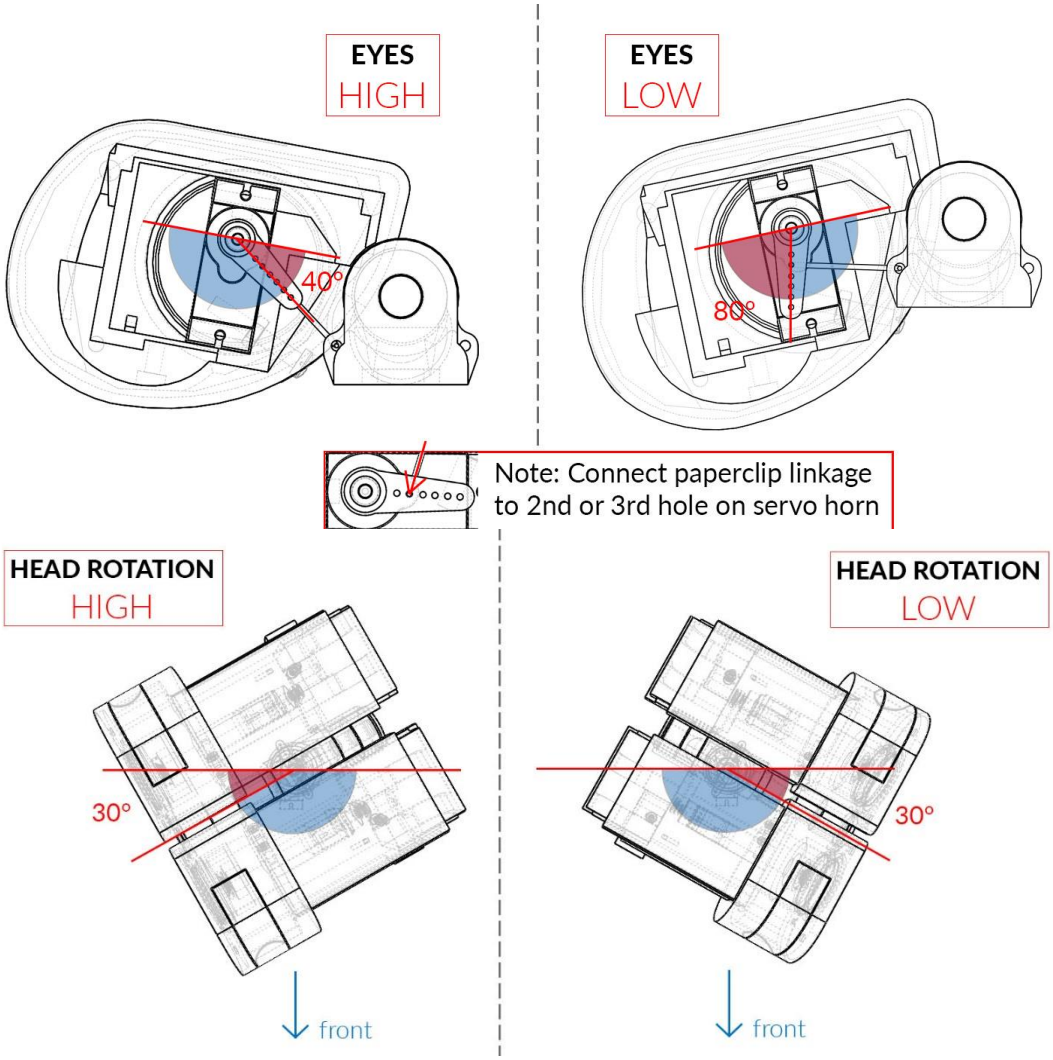
To assemble the robot successfully we need the following materials:

- The 3d parts of the models that were provided The 3D parts where found in the following link: https://www.thingiverse.com/thing:3703555
- At least 5 micro servo motors. (mg90s or sg90)
- 1 PCA9685 module - i2c 16-Channel PWM Servo Motor Driver
- 1 MB102 breadboard power supply module
- 1 breadboard
- 1 raspberry pi 4 model B 8GB
- 1 coral USB accelerator
- 1 Raspberry Pi camera module (preferably 12MP) or any other cam
- Google voice kit or any other microphone and speaker (preferably voice kit due to lack of power)
- 1 12v AC adapter
- 1 mini black hat hack3r module, in case you use google voice kit
- 1 PIO Ribbon Cable 40P for Raspberry Pi, in case you use the google voice kit
- 2 male to male cables
- 5 female to female cables
- 1 raspberry pi camera extension cable
- M3 Bolts 6 mm 10mm and 20 mm for assembling the robot
- 1 screwdriver
- 1 HDMI cable
- 1 monitor
- 1 power cable USB type C for the Raspberry Pi 4
- 1 32GB class 10 micro sd, in order to install the provided image file
- 1 mouse
- 1 keyboard

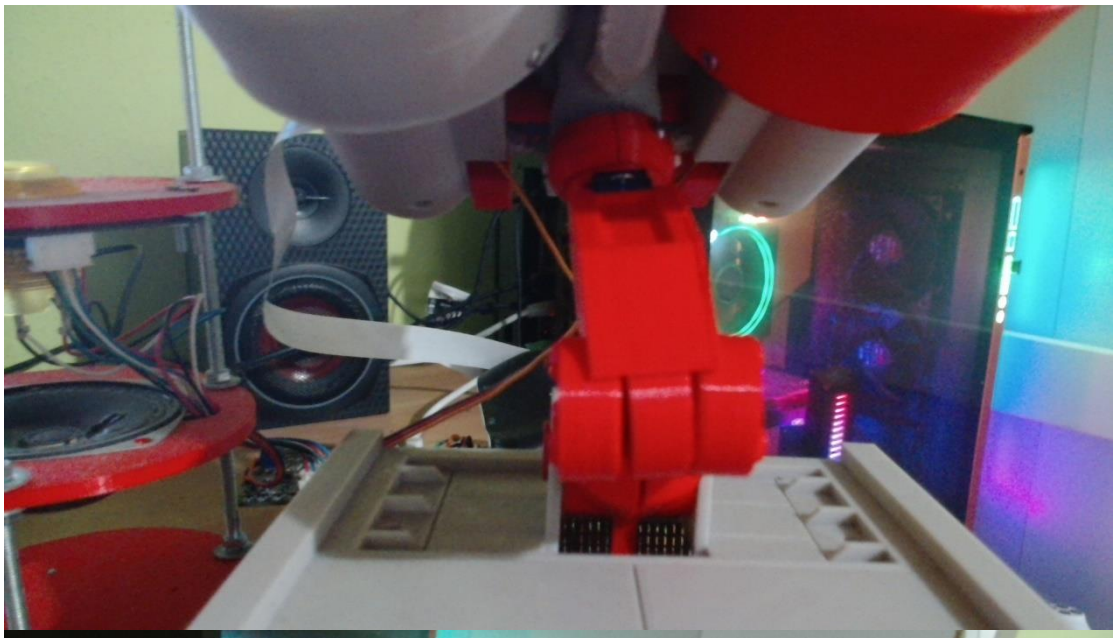
5.2 ASSEMBLING THE ROBOT

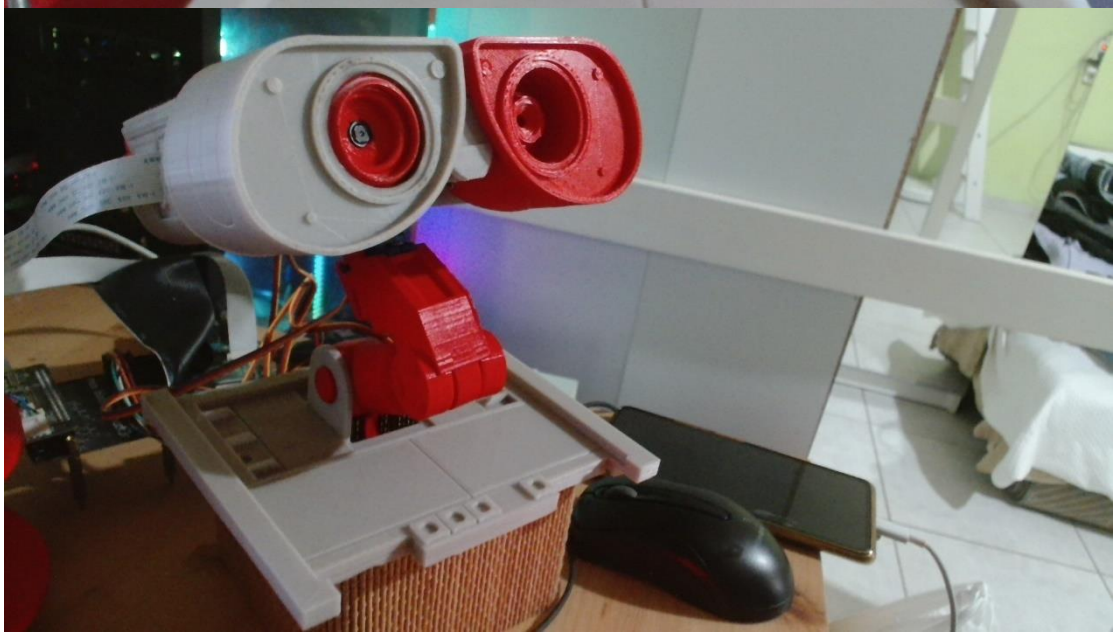
In the images below provided by https://wired.chillibasket.com/3d-printed-wall-e/ ,you will be able to assemble the head of the robot.

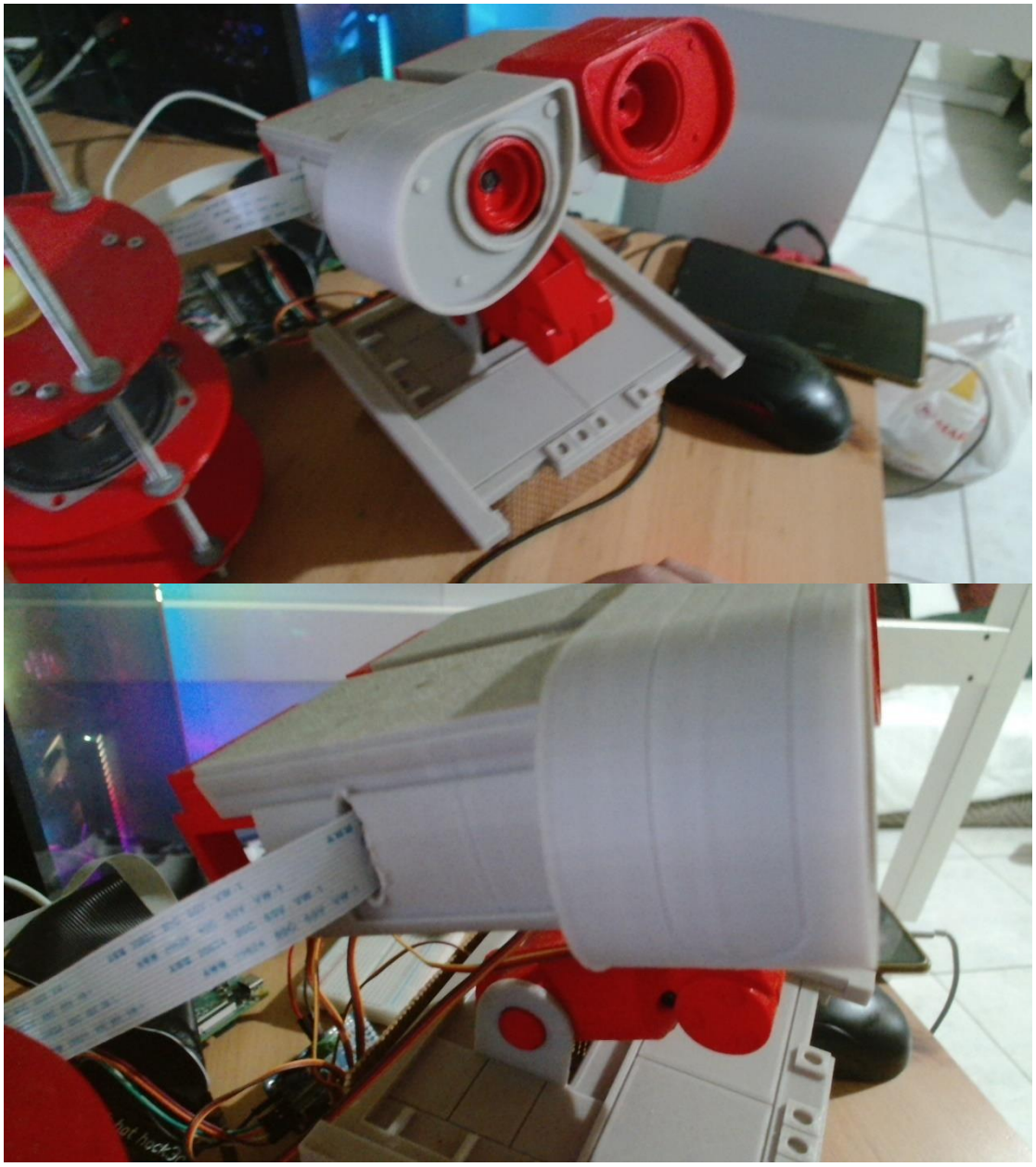


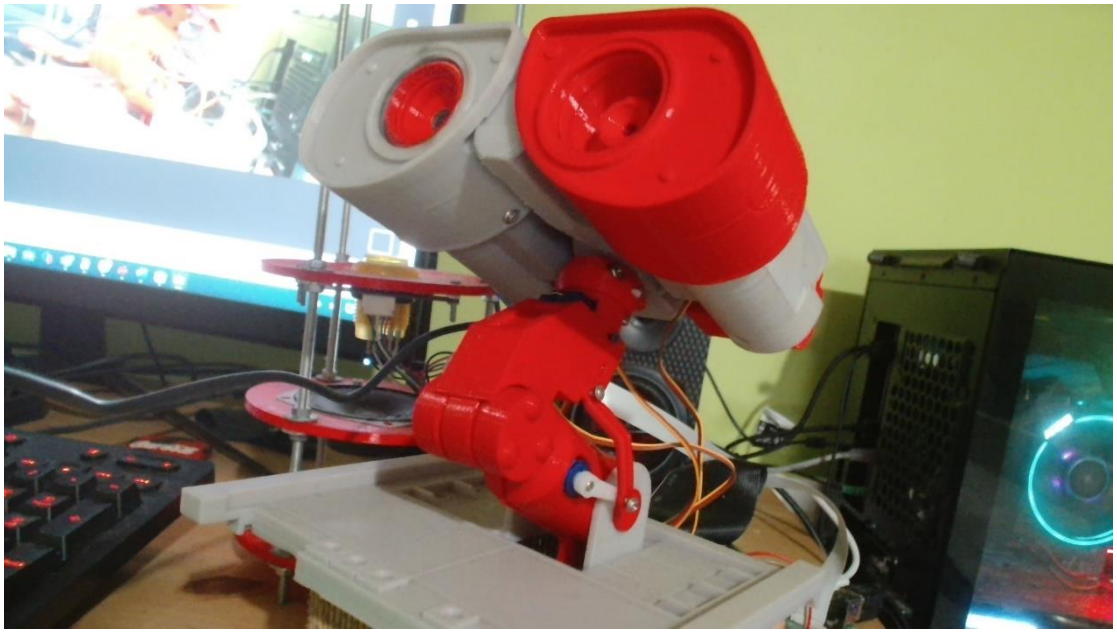

EYES
HIGH
40°
EYES
LOW
80°
Note: Connect paperclip linkage
to 2nd or 3rd hole on servo horn
HEAD ROTATION
HIGH
30°
front
HEAD ROTATION
LOW
30°
front

To expand the functionality of the robot you can attach the Raspicam in one of the robot's eyes.
If you successfully assemble the robot it should look like that:



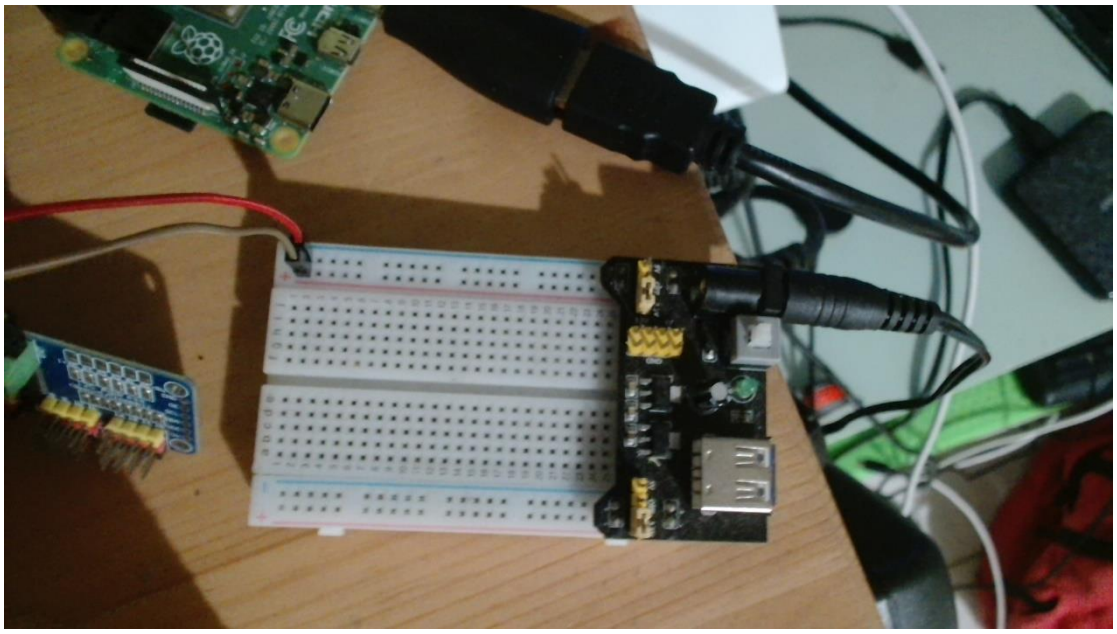




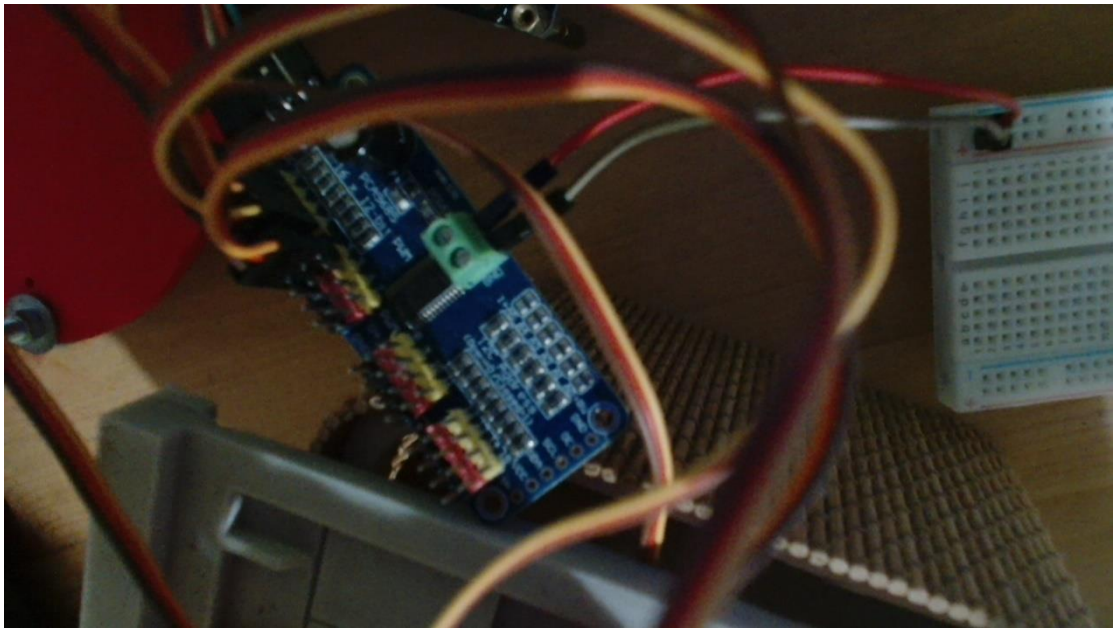


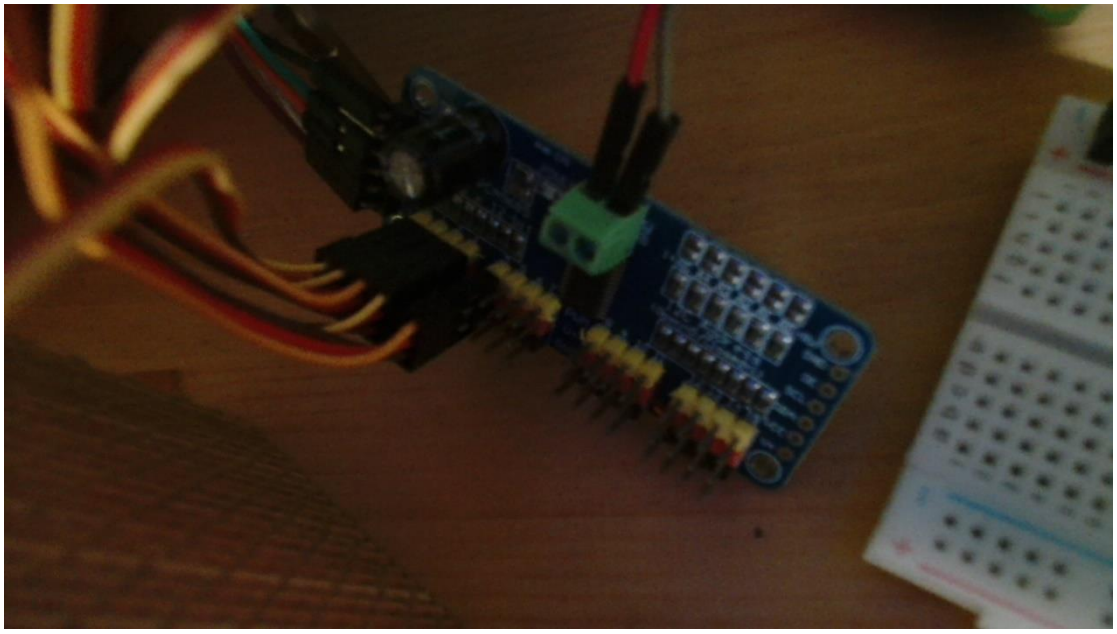
5.3 ASSEMBLING THE ELECTRONICS

Having completed the assemble of the robot, you need to assemble the electronic parts. First you need to connect the MB102 circuit on the breadboard and mount the male to male cables on the breadboard positive and negative pole.

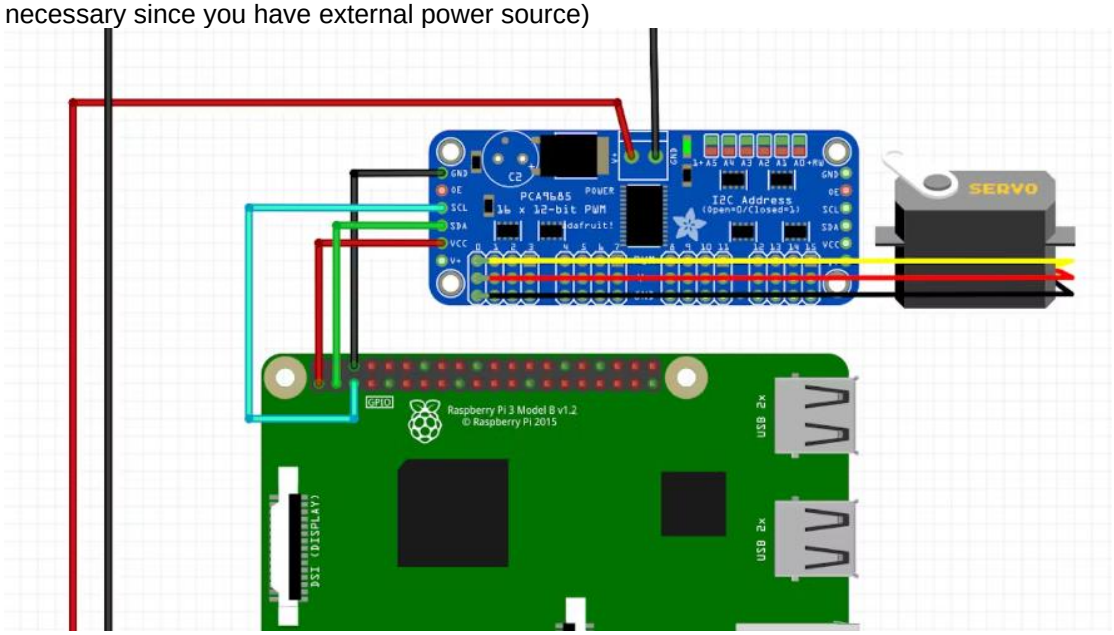


Then you need to connect the other side of the cables to the PCA9685 module, then install the cables from the motors, in the first 5 pinouts of the module. Last step, connect the female to female cables to the side of the module, leaving the OE pin without a cable. The module should look like that:





Then you will need to attach the cables on the raspberry pi. (Note that the 5V pin is not necessary since you have external power source)

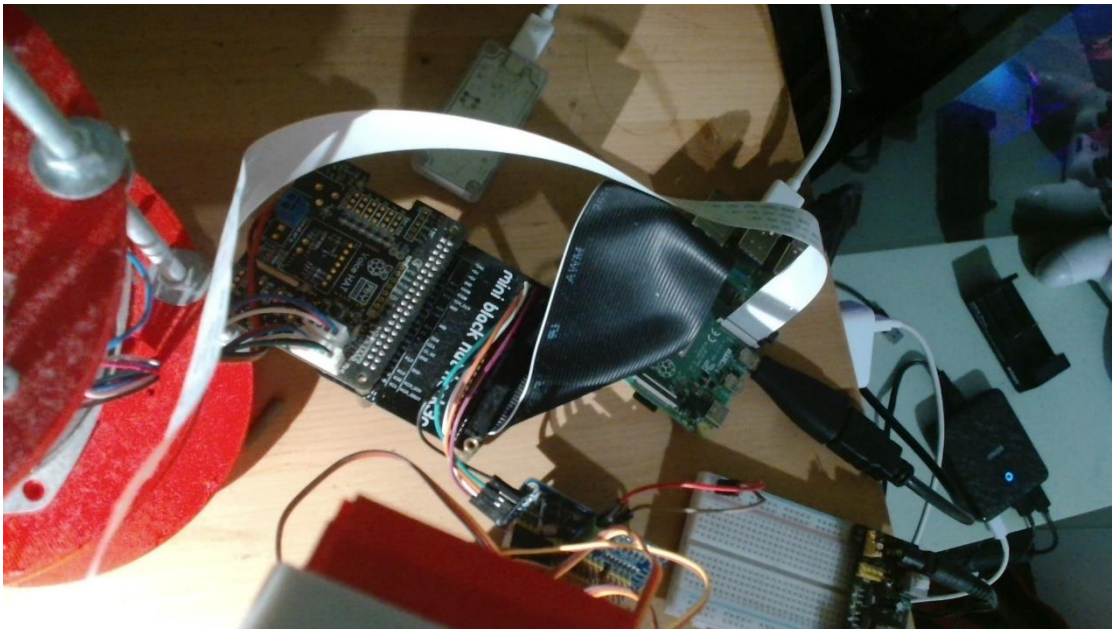


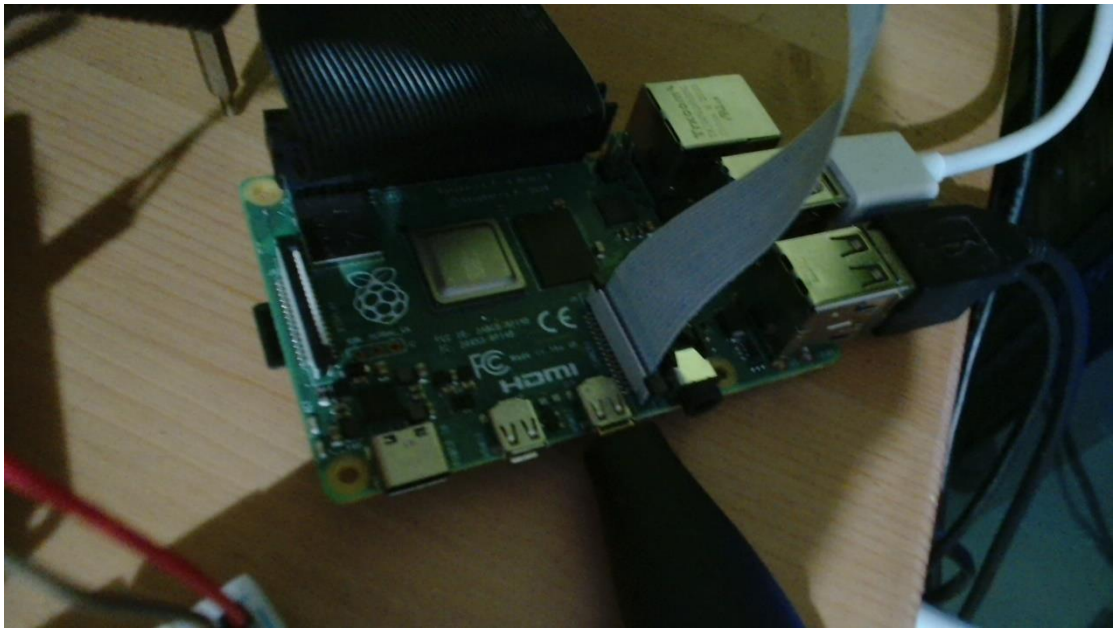
Then you need to format the sd card with the provided image file and insert it in the raspberry or make a clean installation copy the python files and finally import the libraries mentioned in them using a terminal and the pip3 install command.
If you choose the first option, you will need Win32 Disk Imager program to write the image on the sd card.
If you choose the second option, first you will need to execute the following commands in the terminal:

sudo apt-get update

sudo apt-get upgrade

Its important to use low power devices, since the module can be powered with a max of 5A.
In the following images you can see how the module looks with the aiy google voice kit.





Keep in mind that using the google voice kit is recommended, cause of the low power need. One more advantage of its usage is the fact that it does not occupy usb slots. On the usb 2.0 slots you should connect the keyboard and the mouse. On the usb 3.0 slot you must connect the google coral usb accelerator, since it can only be powered through usb3.0 connection.

6 EXPLORING THE CODE

6.1 ROBOT CAPABILITIES

The robot can perform the following actions:

- Object recognition
- Person recognition
- Perform reinforcement learning in order to learn to recognize more people
- Gesture recognition
- Locate and track movement
- Emotion detection
- Count the objects and the persons in its view
- Utilize motors in order to perform movement actions
- Detect if the user is sleepy
- Utilize three different chatbots, ChatGPT, google assistant and Chatsonic, in order to communicate and provide info to user

6.2 CHATGPTPPROJECT.PY

This code demonstrates how to use OpenAI's ChatGPT API to generate responses to questions.

6.3 CHATSONICPROJECT.PY

This code demonstrates how to use Chatsonic API to generate responses to questions.

6.4 MYGOOGLEASSISTANT.PY

This code extends google-assistant-sdk samples so that the user can generate text responses to text questions. Original sample was the textinput.py file provided by Google. Though it needed to be modified heavily so that the generated assistant instance responds with the actual response from the google assistant API and not with the complementary text, as in the example file. Furthermore, I had to change the interface to use Google Assistant API, cause of deprecation of google-assistant-library library, since it was replaced by google-assistant-sdk library. Assistant-helpers.py, Audio-helpers.py, Browser-helpers.py and Device-helpers.py files are also downloaded from google's Git and they are mandatory in order to use Google assistant API.

6.5 DEEPFACEEMOTIONDETECTION.PY

This code defines a class called EmotionDetection that performs emotion detection on video frames using the OpenCV and FER (Facial Expression Recognition) libraries. Here's a brief overview:

The EmotionDetection class initializes by loading a face detection model from the XML file 'haarcascade_frontalface_default.xml' and creating an instance of the FER emotion detector.

The do_emotion_recognition method takes a frame as input, converts it to grayscale, and detects faces using the loaded face detection model. It then extracts each detected face, attempts to recognize the dominant emotion using the FER model, and stores the emotions in a list.

The process_frame method is intended to be called in a multi-threaded environment (using a lock) to process each frame. It invokes the do_emotion_recognition method, passing the frame, and returns the resulting emotions as a list.

Within the do_emotion_recognition method, for each detected face, the corresponding region of interest (ROI) is extracted from the frame. The FER model is applied to the ROI to obtain the dominant emotion and its associated emotion score.

Any exceptions that occur during the face detection or emotion recognition process are caught, and the emotion is set as 'Unknown' in such cases.

The resulting emotions are stored in a list and returned at the end of the do_emotion_recognition method.

In summary, this code sets up a class for performing emotion detection on video frames by utilizing the OpenCV library for face detection and the FER library for emotion recognition. It provides methods to process individual frames and extract emotions from detected faces.

6.6 DrowsinessMediapipe.py

This code defines a class called DrowsinessDetection that uses the OpenCV, NumPy, and MediaPipe libraries to detect drowsiness in a person's eyes. Here's a brief overview:

The DrowsinessDetection class initializes by setting up MediaPipe's face mesh model (mp_face_mesh) for detecting facial landmarks. It also defines lists of indices (RIGHT_EYE and LEFT_EYE) that correspond to the specific landmarks representing the right and left eyes in the face mesh.

The open_len method calculates the vertical distance between the minimum and maximum y-axis points of a given set of landmarks.

The do_detection method takes a frame as input and processes it to detect drowsiness. It converts the frame to RGB format and retrieves the height and width of the frame. It then applies the face mesh model to detect landmarks on the face.

If landmarks are detected, it extracts the landmarks for the right and left eyes based on the predefined indices. It calculates the open length (vertical distance) of each eye.

The frames_num, max_left, and max_right variables are used to keep track of the maximum open length observed for each eye over a certain number of frames. If more than 9 frames have been processed, the counters are reset to avoid false positives due to sudden changes.

If the open length of either eye is less than or equal to half of the maximum open length observed for that eye plus one, it indicates drowsiness. The drowsy_frames counter is incremented.

If the drowsy_frames counter exceeds a certain threshold (1 in this case), it returns 'ALERT' to indicate drowsiness. Otherwise, it returns 'OK'.

If no face is detected in the frame, it returns 'no face'.

The process_frame method is intended to be called in a multi-threaded environment (using a lock) to process each frame. It invokes the do_detection method, passing the frame, and returns the drowsiness result as a string.

In summary, this code utilizes the face mesh model from MediaPipe to detect landmarks on the face, specifically the landmarks corresponding to the right and left eyes. By monitoring the open length of the eyes over multiple frames, it determines if the person's eyes

are showing signs of drowsiness and returns an appropriate alert.

6.7 FACEDETECTOR.PY

This code defines a class called PersonDetection that uses various libraries such as OpenCV, face_recognition, pickle, FaceDetector, and pycoral to perform person detection and recognition. Here's a brief overview:

The PersonDetection class initializes by setting up the paths to the model file (modelPath) and label file (labelPath) for person recognition. It also initializes the TFLite interpreter with the model and loads the labels.

The classifyImage method takes an image and the TFLite interpreter, resizes the image to the input size required by the model, sets the input tensor, runs the interpreter, and retrieves the classification results.

The do_person_recognition method takes a frame as input and processes it to detect and recognize people. It resizes the frame and converts it to RGB format. It then applies the face detection model to detect faces in the frame.

If faces are detected, it iterates over the detected faces and extracts the bounding box coordinates. It crops the face region from the frame and passes it to the classifyImage method for person recognition.

The classifyImage method performs classification on the cropped face image using the TFLite interpreter. If the confidence score of the top prediction is above a threshold (0.7 in this case), it retrieves the corresponding label from the labels file.

If a recognized person's label is above the confidence threshold, it appends the name to the names list and the bounding box coordinates to the boxes list. If the label is below the threshold, an empty string is appended.

The code checks if a recognized face is not in the found_people list, which is used to track previously detected people. If it's a new person, their name is added to the found_people list and the names list.

Finally, the names list is returned, which contains the names of recognized people (or empty strings for unrecognized faces).

The process_frame method is intended to be called in a multi-threaded environment (using a lock) to process each frame. It invokes the do_person_recognition method, passing the frame, and returns the person recognition results as a list of names.

In summary, this code utilizes a face detection model and a person recognition model to detect and recognize people in a video frame. It applies the face detection model to locate faces and crops the face region for person recognition using a TFLite-based model. It then associates recognized names with bounding boxes and tracks new people encountered in subsequent frames. Furthermore, the model is edgetpu optimized, so that it takes advantage of the Google Coral USB Accelerator.

6.8 IMAGECOLLECTOR.PY

This code enables the collection of a person's image dataset for face recognition training. It prompts the user for their name, creates a new folder for the dataset, captures frames from a video stream, detects and crops the face region, and saves the collected images in the corresponding folder.

6.9 IMAGEENCODER.PY

The code performs the process of encoding images and retraining a face recognition model using the Edge TPU. It uses the ImprintingEngine to handle the retraining process, runs inferences on the images using an interpreter, and continues training classes with their respective image embeddings. The retrained model is then saved for future face recognition tasks. The starting model for this operation is mobilenet_v1_1.0_224_l2norm_quant_edgetpu.tflite. This model is used for image classification problems. I figured out that person recognition, is a classification problem, if you are able to cut out the face of the person from the frame. To do so I used in the FaceDetector and the ImageCollector file Mediapipe's face detection solution and used opencv to cut the detected face out of the frame. Furthermore, the model is edgetpu optimized, so that it takes advantage of the Google Coral USB Accelerator.

6.10 GESTURETRACKING.PY

This code defines a handDetector class that utilizes the Mediapipe library to detect and track hand landmarks in real-time video frames. It also provides methods to find the hand landmarks and count the number of fingers extended. Additionally, there is a function do_recognize_gesture_RPS that utilizes the handDetector class to recognize hand gestures for a rock-paper-scissors game.

The handDetector class initializes with parameters such as mode, maximum number of hands, and confidence thresholds. It uses the mpHands.Hands class from the Mediapipe library to create an instance for hand detection and tracking. It also imports mpdraw for drawing the landmarks on the image.

The findHands method takes an image as input, converts it to RGB format, and processes it using the Hands object. It draws the detected hand landmarks on the image if the draw parameter is set to True.

The findPosition method takes an image, hand number (default is 0), and draw flag. It returns a list of landmarks for the specified hand, where each landmark contains the landmark ID and its corresponding x and y coordinates.

The do_recognize_gesture_RPS function performs gesture recognition for the rock-paper-scissors game using the hand detection and tracking capabilities provided by the handDetector class.

Inside the function, it initializes the hand detector, sets up the video capture, and sets the desired width and height for the captured frames.

A while loop is used to continuously read frames from the video capture.

The captured frame is flipped vertically and passed to the findHands method to detect and draw hand landmarks.

The findPosition method is then called to obtain the landmarks' positions.

The function Right_hand checks if the hand is the right hand and counts the extended fingers using the positions of the landmarks. Similarly, Left_hand does the same for the left hand.

The total count of fingers for both hands is calculated.

Based on the finger count, the corresponding gesture is assigned (rock, paper, scissors), and the loop is broken.

The recognized gesture is returned by the function.

In summary, the code utilizes Mediapipe to detect and track hand landmarks in real-time video frames. It then counts the number of extended fingers and recognizes hand gestures for the rock-paper-scissors game.

6.11 GesturesBonusAction.py

This code utilizes the mediapipe library along with OpenCV (cv2) to track hand movements through a webcam.

The required libraries, cv2 for computer vision and mediapipe for hand tracking, are imported.

The mediapipe hand tracking model is initialized, and the webcam capture is set up.

A frame is captured from the webcam and flipped vertically to match the user's viewpoint.

The width, height, and channels of the frame are extracted.

Variables to store the previous maximum hand coordinates and the hand direction are initialized.

A function is defined to calculate the hand position based on the current and previous coordinates.

An infinite loop begins to continuously process frames from the webcam.

A new frame is read, flipped horizontally, and converted to RGB format.

The hand tracking model processes the frame to detect hand landmarks.

If hand landmarks are detected, each detected hand is iterated over to find the maximum and minimum coordinates.

A rectangle is drawn around the hand region, and the hand landmarks are visualized on the frame.

The hand direction is determined by comparing the current and previous maximum coordinates.

The previous maximum coordinates are updated with the current maximum coordinates.

The hand direction is displayed as text on the frame.

The frame, showing the annotated hand and the hand direction, is displayed, and the code waits for a key press to exit.

To summarize, this code utilizes mediapipe and OpenCV to track hand movements in real-time using a webcam, displaying the direction of the detected hand on the screen.

6.12 MotionControl.py

This code utilizes adafruit_servokit library to perform all the actions associated with the servo motor movements. The servos used are 5 sg90 mini servo motors that have a motion range from 0 to 180. The file contains all the moves that the robot can perform on occasion.

6.13 Movement_recognition.py

This code performs motion recognition using a webcam and OpenCV (cv2). Here are the key steps:

The necessary libraries and a custom module called motionControl are imported. The motionRecognition function is defined, which takes the number of frames (FrNum) to process. The webcam capture is initialized, and the width and height of the captured frames are obtained. The frame is divided into 8 parts for motion analysis. An array is created to store the previous frame.

Inside a loop, frames are read from the webcam and processed until the desired frame count is reached. The frame is flipped vertically, and if there is no valid frame, the loop breaks and is converted to grayscale.

The absolute difference between the current frame and the previous frame is computed. Thresholding and dilation operations are applied to obtain a binary motion mask. Contours are detected in the dilated image. Each part of the frame is analyzed for motion by dividing it into sub-regions. The motion area in each part is calculated, and the maximum motion in each direction is updated. The general position of the movement is determined based on the maximum motion in each direction. Depending on the detected movement position, corresponding actions are triggered using the motionControl module.

The previous frame is updated with the current grayscale frame.

After processing the desired number of frames, the webcam capture is released, and all windows are closed.

In summary, this code performs motion recognition by analyzing the differences between consecutive frames from a webcam, and based on the detected motion direction, it triggers corresponding actions using the motionControl module.

6.14 ObjectRecognitionCoral.py

The provided code sets up a class called ObjectDetection for performing object detection using the EfficientDet-Lite3 model with an Edge TPU accelerator. Here's a brief explanation of the code:

The code imports necessary libraries including time, OpenCV (cv2), NumPy (numpy), and the PIL library (PIL) for working with images.

It imports the required packages from the Coral library for working with the Edge TPU.

The ObjectDetection class is defined, which initializes the model and label paths, sets up the Coral interpreter, and loads the labels from a file.

The do_object_recognition method performs object recognition on a given frame.

It preprocesses the frame by converting it to RGB color space and resizing it to match the model's input tensor shape.

The preprocessed frame is converted to a PIL Image and passed to the interpreter for object detection.

Detected objects with a score higher than 0.5 are extracted and their labels, bounding box coordinates, and scores are processed.

The method returns a list of recognized object labels.

The process_frame method, which takes a frame and a lock, is defined to be used in a multithreaded environment.

Within the method, the do_object_recognition method is called to perform object detection on the frame, and the result is returned.

In summary, the code sets up an object detection class using the EfficientDet-Lite3 model with the Edge TPU. It provides methods for performing object recognition on frames and processing the results.

6.15 RPSGame.py

The code implements a rock-paper-scissors game with gesture recognition. It prompts the user to make a gesture, recognizes it, compares it with the computer's choice, determines the winner, performs corresponding moves using motion control, and allows the player to play again if desired.

6.16 ROBOT.PY

This is the program that orchestrates the functionality of the robot.

The program captures frames from a camera and processes them concurrently using multiple threads. Each thread is responsible for a specific task like emotion recognition, object detection, drowsiness detection, and person detection. The results of these tasks are stored in different lists.

Based on the detected objects, emotions, and queries from the user, the program takes specific actions. For example, it can count objects in the frame, recognize and greet people, detect dominant emotions, play a game of rock-paper-scissors, or interact with Google Assistant or other chatbot models like ChatGpt and ChatSonic.

The program continuously listens for user queries and performs the corresponding actions based on the recognized commands. The execution flow includes motion recognition, where the program checks for specific movements, and if the user requests a change in the assistant, the program switches to a different assistant module accordingly.

Overall, this code integrates computer vision and natural language processing functionalities to create an interactive system capable of performing various tasks and assisting users based on their queries and the analyzed visual input from the camera.

6.17 CODE Paths

Inside the folder PiRobot, you will find the final code that was used to implement the robot functionalities.

Inside the folder Python-Standalone-code, you will find the standalone code for each functionality that was implemented for the needs of the code, implemented in two ways. One that uses pycoral libraries, to use Coral USB Accelerator and one without it. Also you will find test programs of other functionalities, like sound direction detection and a hand tracking system.

7 GOOGLE ASSISTANT VS CHATGPT VS CHATSONIC

7.1 GOOGLE Assistant

Google Assistant is a virtual assistant developed by Google that is designed to provide a wide range of capabilities and help users with various tasks. Here are some of the key capabilities of Google Assistant:

Voice Commands and Natural Language Understanding: Google Assistant is primarily activated by voice commands and can understand natural language input. Users can ask questions, give commands, and have conversations with the assistant in a more natural and intuitive manner.

General Knowledge and Information: Google Assistant is connected to Google's vast knowledge database, which enables it to answer a wide range of general knowledge questions. It can provide information about facts, figures, historical events, current events, sports, weather forecasts, and much more.

Personalized Assistance: Google Assistant can be personalized to suit individual users' needs. It can access and provide information specific to the user, such as calendar events, reminders, flight information, package tracking, and personalized recommendations based on user preferences.

Smart Home Control: Google Assistant can integrate with various smart home devices and platforms, allowing users to control their connected devices using voice commands. Users can control lights, thermostats, smart locks, security cameras, and other compatible devices hands-free.

Multimedia Playback: Google Assistant can play music, podcasts, audiobooks, and other media content on compatible devices. Users can request specific songs, artists, albums, or playlists, and the assistant can stream the content from various online sources or access locally stored media.

Navigation and Directions: Google Assistant can provide directions and navigation assistance. Users can ask for directions to a specific location, inquire about nearby places of interest, and get real-time traffic updates.

Communication and Messaging: Google Assistant enables users to make calls, send text messages, and initiate video calls using voice commands. It can also read and compose emails, schedule appointments, and manage contacts.

Translation and Language Support: Google Assistant supports translation of words, phrases, and sentences into different languages. It can act as a language translator and help users communicate in foreign languages.

Task Automation and Integration: Google Assistant can integrate with various third-party apps and services to automate tasks and enhance productivity. Users can set up routines and shortcuts to perform multiple actions with a single command.

These are just some of the capabilities of Google Assistant. Over time, Google continues to enhance and expand the assistant's functionality, adding new features and integrations to provide users with a comprehensive and personalized virtual assistant experience.

7.2 Google Assistant Cons

The cons of Google Assistant API are:

Limited Customization: While the Google Assistant API offers a range of features and capabilities, the level of customization available to developers is somewhat limited. Developers may not have full control over the behavior or responses of the Google Assistant, which could restrict the ability to tailor the user experience according to specific requirements.

Dependency on Internet Connectivity: The Google Assistant API requires an active internet connection to function properly. This means that applications or devices using the API may become non-functional or limited in their capabilities when offline or in areas with poor internet connectivity. This dependency on internet connectivity can be a significant drawback in certain use cases.

Privacy and Data Concerns: When using the Google Assistant API, user interactions and data are transmitted to Google for processing and analysis. This raises privacy concerns for some users who may be apprehensive about sharing their personal information or conversations with a third party. It is important for developers to ensure that appropriate privacy and data protection measures are implemented when working with the Google Assistant API.

Limited Platform Compatibility: While the Google Assistant API is widely supported across various devices and platforms, there may still be some limitations or restrictions on compatibility. Developers may need to consider the specific requirements and limitations of each platform or device they intend to target, which could add complexity to the development process.

Development and Maintenance Effort: Integrating the Google Assistant API into an application or service requires dedicated development effort and ongoing maintenance. This includes staying up to date with API changes and updates from Google, ensuring compatibility with evolving platforms and devices, and addressing any potential issues or bugs that may arise.

It's important to consider these cons alongside the pros when deciding to use the Google Assistant API, as they may impact the suitability of the API for specific use cases.

7.3 CHATGPT

ChatGPT, powered by OpenAI's GPT-3.5 architecture, offers a range of capabilities for natural language understanding and generation. Here are some key features and capabilities of ChatGPT:

Conversational Assistance: ChatGPT is designed to engage in human-like conversations and help on various topics. It can understand and respond to user queries, helping with information, recommendations, explanations, and more. Users can have interactive and dynamic conversations with ChatGPT as if they were conversing with a human.

Information Retrieval: ChatGPT can fetch information from the web and provide users with answers to factual questions. It can access a vast amount of knowledge and provide relevant and up-to-date information on a wide range of topics, including general knowledge, current events, historical facts, scientific concepts, and more.

Creative Writing: ChatGPT has a creative side and can generate text in a coherent and imaginative manner. It can assist with creative writing tasks, such as generating stories, poems, dialogues, or even brainstorming ideas for creative projects. Users can collaborate with ChatGPT to explore creative possibilities.

Language Translation: ChatGPT can translate text from one language to another. Users can input sentences or phrases in one language and receive translations in their desired language. This feature facilitates cross-lingual communication and helps users overcome language barriers.

Text Summarization: ChatGPT can summarize lengthy texts or documents into shorter, more concise versions. It can extract key points, important details, or main ideas from a given text, making it easier for users to grasp the essence of the content without having to read through the entire document.

Language Learning and Practice: ChatGPT can assist users in learning and practicing different languages. It can provide translations, help with grammar and vocabulary, offer language learning tips, and engage in language practice exercises. Users can enhance their language skills through interactive conversations with ChatGPT.

Personal Assistant Functions: ChatGPT can perform tasks like setting reminders, scheduling events, providing weather updates, calculating mathematical expressions, and more. It can act as a virtual personal assistant, helping users with day-to-day tasks and providing relevant information and assistance.

General Knowledge and Trivia: ChatGPT has access to a wide range of general knowledge and can engage in trivia questions and discussions. Users can ask questions, seek interesting facts, explore topics of interest, and have engaging conversations with ChatGPT on various subjects.

It's important to note that while ChatGPT is a powerful language model, it may occasionally generate inaccurate or misleading information. Users should exercise critical thinking and verify information obtained from ChatGPT from reliable sources when necessary.

7.4 CHATGPT CONS

Here are some cons of using ChatGPT or similar language models:

Lack of Common Sense: ChatGPT lacks true understanding of the world and does not possess common sense reasoning. It generates responses based on patterns it has learned from training data, which can sometimes lead to nonsensical or incorrect answers. It may

struggle to provide accurate information or contextually appropriate responses in certain situations.

Overconfidence and Incorrect Information: ChatGPT can sometimes generate responses with a high level of confidence, even when the information provided is inaccurate or false. Users need to critically evaluate the responses and verify the information independently, as the model does not have the ability to fact-check or verify the accuracy of its own responses.

Sensitivity to Input Phrasing: The way a question or prompt is phrased can significantly affect the quality of the response generated by ChatGPT. Small changes in wording or structure may result in different or inconsistent answers. Users may need to experiment with different phrasings or provide more explicit instructions to get the desired response.

Bias and Inappropriate Content: Language models like ChatGPT can inadvertently exhibit biases present in the training data, leading to biased or unfair responses. It may also generate inappropriate or offensive content in certain cases. Efforts have been made to mitigate these issues, but biases and inappropriate content can still arise and require ongoing monitoring and improvement.

Lack of Emotional Understanding: ChatGPT does not have emotional understanding or empathy. It may not respond appropriately to emotionally charged or sensitive topics. It is important for users to remember that ChatGPT is an AI language model and lacks the emotional intelligence and human-like understanding that humans possess.

Dependence on Training Data: ChatGPT learns from the data it is trained on, which means it can potentially inherit the biases, inaccuracies, or limitations present in the training data. It may not be able to provide reliable information on topics that are underrepresented or poorly covered in its training data.

Ethical Considerations: The use of ChatGPT and similar models raises ethical considerations, such as the responsible handling of user data, the potential for misuse or malicious intent, and the impact on human labor and employment. Care must be taken to ensure the appropriate and ethical use of AI language models.

These cons highlight the limitations and challenges associated with ChatGPT and similar models. It is important to set realistic expectations and use these models responsibly, understanding their strengths and weaknesses.

7.5 ChatSonic

ChatSonic is an AI-powered chatbot developed by WriteSonic that offers a range of powerful capabilities to assist users in generating unique and non-plagiarized content. Leveraging natural language processing (NLP) technology, ChatSonic is designed to enhance the writing process by providing creative suggestions, generating text, and offering up-to-date information on current events. Here are some of the notable capabilities of ChatSonic:

AI-powered Text Generation: ChatSonic utilizes advanced AI algorithms to generate text quickly and efficiently. It can engage in natural and dynamic conversations, providing well-thought-out responses that are tailored to the user's input. The AI model behind ChatSonic has been trained on a vast amount of text data, enabling it to generate high-quality content across various topics.

Text-to-Art Generation: One of the standout features of ChatSonic is its ability to generate art from text descriptions. Users can describe the type of art they want to see, and ChatSonic will produce a unique piece of AI-generated art based on those descriptions. This integration of text generation and art creation sets ChatSonic apart, making it a versatile tool for content creators and artists.

Natural Language Understanding: ChatSonic exhibits an advanced understanding of natural language, allowing for smooth and conversational interactions. It can comprehend and interpret user queries, prompts, and requests accurately. This natural language understanding enables a more intuitive and human-like conversational experience with the chatbot.

Content Creation: ChatSonic is a valuable tool for generating various types of written content. It can assist users in crafting blog posts, song lyrics, jokes, weather predictions, video transcripts, essays, and more. By providing creative suggestions and generating relevant text, ChatSonic serves as a helpful writing companion, especially when inspiration is needed.

Up-to-date Information on Current Events: Unlike some AI models that rely on static data from the past, ChatSonic has access to the latest information and trends. It can provide insights and updates on current events, including news, sports, and other timely topics. This feature ensures that users stay informed and can engage in discussions or create content that reflects the most recent developments.

Cost-effective Automation: While ChatSonic is a paid service, it offers a cost-effective pricing model compared to some other AI writing tools. The additional features, such as image generation, text-to-art capabilities, and up-to-date information, make ChatSonic a valuable investment for users seeking advanced AI writing capabilities. The affordability of the tool makes it accessible to a wide range of individuals and businesses.

It's important to note that while ChatSonic is a powerful tool, it has its limitations. As with any AI technology, there may be occasional inaccuracies or the need for human intervention to ensure the final content meets desired standards. However, ChatSonic's capabilities can significantly enhance the writing process, spark creativity, and provide valuable assistance for content creators, writers, and individuals looking to generate unique and engaging text.

Overall, ChatSonic's combination of AI-powered text generation, text-to-art capabilities, natural language understanding, content creation support, and up-to-date information make it a versatile and valuable tool for anyone looking to enhance their writing and creative processes.

7.6 ChatSonic Cons

Here are some potential cons of ChatSonic:

Word Limit: ChatSonic has a word limit for both the free trial and paid plans. The limited word count may not be sufficient for power users, such as bloggers, freelancers, or marketers, who need to process large amounts of text. The starting price of $19 offers a word limit of 15,000 words, which may not be enough for certain users.

Issues with Image Generation: ChatSonic's image generation feature may be challenging to use effectively. Describing images in detail with text may not always yield the desired results. The feature's performance may not meet expectations, and users may have to pay extra for generated images. Additionally, if ChatSonic fails to generate the requested image, users still incur a cost without receiving the desired content.

Lack of Knowledge: ChatSonic's training is based on information available until 2021. If users do not enable the option to include the latest Google data, ChatSonic may not be aware of current issues and information. Enabling this option incurs an additional word fee, limiting access to up-to-date knowledge without additional charges.

Factual Mistakes: Like any AI writing tool, ChatSonic is not infallible and may make factual mistakes. Users should be aware that the suggestions and corrections provided by ChatSonic may not always be accurate, requiring additional fact-checking or editing.

Human Intervention Required: While ChatSonic is a useful tool, it should not be seen as a complete replacement for human writers. Human intervention is necessary to ensure the quality and coherence of the writing produced by ChatSonic.

Repetition: ChatSonic may suggest the same corrections or ideas multiple times, leading to repetition. This repetition can be frustrating and may hinder the overall efficiency and effectiveness of the writing process.

It's important to note that these cons are based on the provided information and may not encompass all potential limitations or issues associated with ChatSonic. For a

comprehensive evaluation, it's advisable to consult official documentation or reviews specific to ChatSonic.

7.7 Which One Wins As A Home Assistant?

Google Assistant API is widely integrated and offers voice control, but customization options may be limited, and privacy concerns can arise due to data collection.

ChatSonic excels in natural language understanding and customer support, simplifying interactions. However, it has limited knowledge and updates, word limits, image generation issues, and requires extra payment for up-to-date information.

ChatGPT showcases high language proficiency, works offline, and has general-purpose AI capabilities. However, its knowledge is only up to Q3 2021, lacks domain specialization, and may generate inaccuracies.

The choice depends on specific requirements and preferences, considering factors like integration, customization, natural language understanding, customer support focus, knowledge currency, word limits, image generation, language proficiency, and domain specialization.

In the case of the robot, Google Assistant was more helpful regarding news and home automation, while CHATGPT offered the most believable dialog and a great task list creation capability. In the case of Chatsonic, heavy testing couldn’t be completed since it is a bit pricey. Also the API provided for integration with python, responds with the actual response and links from the source that the response was taken, making it difficult to create believable dialogs.

In conclusion a mix of the capabilities of ChatGPT and Google Assistant API was the best approach for the needs of this robot.

8 GOOGLE CORAL USB ACCELERATOR

8.1 WHAT IS IT?

The Google Coral USB Accelerator is a small hardware device developed by Google that provides a high-performance edge computing solution. It is designed to accelerate machine learning (ML) inference tasks on edge devices such as laptops, desktops, and single-board computers.

The Coral USB Accelerator features a USB interface for easy connectivity and compatibility with various devices. It incorporates a custom-built Tensor Processing Unit (TPU), which is a specialized hardware accelerator designed specifically for ML workloads. The TPU delivers fast and efficient ML inference performance, allowing devices to execute ML models with low latency and high throughput.

By offloading ML processing to the Coral USB Accelerator, devices can perform real-time inference tasks locally without relying on cloud services, enabling faster response times, improved privacy, and reduced dependency on internet connectivity. This makes it particularly useful for applications that require quick and reliable ML inference on edge devices, such as robotics, smart home devices, surveillance systems, and more.

The Coral USB Accelerator is compatible with popular ML frameworks like TensorFlow Lite, making it easy to deploy and run pre-trained ML models. It provides a convenient and accessible solution for developers and enthusiasts to harness the power of ML inference in edge computing scenarios, opening up possibilities for advanced AI applications on a wide range of devices.

8.2 USAGE IN THE ROBOT

In terms of the robot, the heaviest tasks where the person recognition task and the object detection task. Both where written and tested, using coral usb accelerator and without it.

8.3 PERSON RECOGNITION

For the person recognition without coral scenario, I used faced encoding to turn the face images into numbers. Once I had all the faced encoded, extracting the facial features real-time and searching the encoding for a much was a fast and in general reliable solution giving me 20fps. But when I had to recreate the encodings file the process was slow and needed almost 1 minute per 30 images it was processing. For the person recognition with coral I used mobilenet_v1_1.0_224_l2norm_quant_edgetpu.tflite as a base model and retrained it to do image classification for person recognition. I used coral for the retraining process and it is now completed in 3 seconds per person it can recognize with 60 images as training data per person. Furthermore, person recognition in live feed runs a 30 fps.

8.4 OBJECT DETECTION

For the object detection without coral scenario, I used the YOLOv6 object detector loading the YOLOv6 model and I processed the frames from the camera to detect the objects in them. In general, I could not achieve more than 5 to 10 fps, depending on the objects it detected on every frame.

On the second implementation I used EfficientDet-Lite3 object detection model with the Edge TPU using the coral package. I processed the frames and performed object detection

using the Coral interpreter and as a result, I could achieve 20 to 25 fps. This significant upgrade helped me perform a great decrease in the time I needed to perform those actions.

8.5 Conclusion

In general Coral USB Accelerator, is a great upgrade for low end devices like the raspberry pi, with which you can perform machine learning tasks in a cost efficient way.

9 FURTHER IMPROVEMENTS

9.1 Ideas To Further Improve The Robot

I order to further improve the robot one could implement:

Real-time Object Tracking: Implement object tracking algorithms to track objects across frames. This can help your robot maintain a continuous focus on a specific object of interest.

Adding rechargeable power supply: Adding rechargeable power supply would help level up the robot and it would open up new possibilities for further development.

Adding wheels: adding wheels on the robot and implementing algorithms for floor mapping and autonomous charging.

Floor mapping: Implement simultaneous localization and mapping (SLAM) algorithms to enable your robot to create a map of its environment and autonomously navigate through it. This can be useful for tasks such as autonomous exploration.

Autonomous charging: Implement a self-charging system for your robot, where it can autonomously navigate to a charging station when its battery is low and resume its tasks once charged. This ensures the robot's autonomy and reduces the need for manual intervention.

Cloud Connectivity: Enable your robot to connect to the cloud for accessing external resources, such as advanced machine learning models, data storage, or remote control and monitoring. Cloud connectivity can expand the capabilities and flexibility of your robot.

Machine Learning based orchestration of the functionalities: Using reinforcement learning, orchestrate the functionalities to be executed on meaningful ways for a human.

Sound direction detection: Using microphone array module, implement sound direction detection. This can help the robot turn its head to the person that speaks to it.

10 IMPLEMENTATION ISSUES

10.1 Major Issues

During the implementation of the code, I run into the following major problems:

Sound direction detection wasn't possible using KY038 modules. These modules only recognize loud noises, being practically useless, to implement the desired functionality.

Emotion detection using custom tensorflow lite model, was not successful. The only available dataset is the FER 2013 dataset, which contains low quality black and white images, while Google Coral classification interpreter, only accept RGB images. To overcome this issue, FER library was used for emotion detection.

Using the models though cloud required additional free space and constant internet connection, making it impossible to keep 3 different implementations and being efficient using a 32GB sd card (maximum capacity for Raspberry Pi 4).

10.2 Minor Issues

During the implementation of the code, I run into the following minor problems:

Raspberry Pi didn't have enough computational power to retrain an object detection model, so I stuck with pretrained models.

I could not successfully synchronize the code to run the object detection and person recognition actions, while the system was taking photos of the user to retrain the person recognition model, in case of detecting an unknown person inside the field of vision. To overcome this issue, when an unknown person appears, the other functionalities stop and the imageCollector action starts.

It is not possible to use the speaker from 2 different thread simultaneously. Also, using a queue, filled with the messages, made the conversation seem inhuman. So, I kept a linear approach to perform the actions that require the speaker.

11 CONCLUSION

In concluding this thesis, I would be remiss not to express my heartfelt gratitude to the individuals who have played a pivotal role in shaping my journey and supporting me throughout this endeavor.

First and foremost, I extend my deepest appreciation to my supervising professor, Themistoklis Panagiotopoulos, for his unwavering guidance, expertise, and invaluable mentorship. His profound knowledge and insightful feedback have been instrumental in refining my research and pushing the boundaries of innovation. I am immensely grateful for his patience, encouragement, and dedication to nurturing my intellectual growth.

I am indebted to my parents, Dimitris and Efi, for their unending love, unwavering belief in my abilities, and unconditional support. Their constant encouragement, sacrifices, and words of wisdom have been the pillars of strength that propelled me forward. Their unwavering faith in my capabilities has been a constant source of inspiration throughout this journey.

To my life partner, Ioanna, I extend my deepest appreciation for her unwavering support, understanding, and encouragement. Her presence, love, and belief in my dreams have been the driving force behind my perseverance. Her unwavering support and willingness to listen have been a source of solace and motivation during both the highs and lows of this research.

Throughout this thesis, we have embarked on a journey to explore the capabilities of an intelligent robotic system designed to enhance human-robot interaction. By integrating state-of-the-art technologies such as computer vision, machine learning, and natural language processing, we have witnessed the evolution of a versatile and engaging companion capable of perceiving, recognizing, and responding to its environment and human counterparts. The results obtained from our research demonstrate the potential of the developed robotic system to revolutionize various domains.

As we conclude, we acknowledge that the field of human-robot interaction is ever-evolving, and there are numerous avenues for further exploration and refinement. Future research endeavors could focus on enhancing the robot's cognitive capabilities, refining its understanding.

12 REFERENCES

- https://www.thingiverse.com/thing:3703555
- https://wired.chillibasket.com/3d-printed-wall-e/
- https://chat.openai.com/
- https://writesonic.com/chat?ref=chatsonic66&fbclid=IwAR2MCEq1qNGqhmbZqFddU0MtOXgCnbxQ5i2zujgzj27gYKuWwvlobsxS4D4&gclid=CjwKCAjwhJukBhBPEiwAniIcNbpyMzokA10LaprVm2sht0pIIxZ3qAzxDhLx30QnXfcnH18xQCLjBxoCtowQAvD_BwE
- https://developers.google.com/assistant/sdk/reference/rpc
- https://coral.ai/examples/
- https://developers.google.com/mediapipe/solutions/guide
- https://github.com/mayurmadnani/fer
- https://docs.opencv.org/4.x/d9/df8/tutorial_root.html
- https://www.tensorflow.org/resources/models-datasets