Logo, company name

Description automatically generated**LEBANESE AMERCIAN UNIVERSITY**

**MCE411– Mechatronic System Design 2**

**Doctor Noel Maalouf**

**Project report**

**Rami Hjeij: #ID 202001988**

**Hassan Al Hakim #ID 202000443**

**Naja Bou Akar #ID 202000174**

Date of Submission: 5/10/2023

Contents

[Figures: 3](#_Toc134637058)

[Methodology: 4](#_Toc134637059)

[Design: 4](#_Toc134637060)

[Robotic arm: 4](#_Toc134637061)

[Conveyor belt: 4](#_Toc134637062)

[Simulation: 10](#_Toc134637063)

[Belt: 10](#_Toc134637064)

[Robotic arm: 12](#_Toc134637065)

[Implementation: 16](#_Toc134637066)

[Process of sorting: 16](#_Toc134637067)

[Problems in the testing phase: 17](#_Toc134637068)

[Results and Limitations: 18](#_Toc134637069)

[Limitations: 18](#_Toc134637070)

[Results: 19](#_Toc134637071)

[References: 20](#_Toc134637072)

# Figures:

[Figure 1:MPU6050 5](#_Toc134637035)

[Figure 2: High Torque 12V DC Motor 6](#_Toc134637036)

[Figure 3: MG996R Servo Motor 6](#_Toc134637037)

[Figure 4: SG90 9g Servo Motor 6](#_Toc134637038)

[Figure 5: Raspberry Pi 4 7](#_Toc134637039)

[Figure 6: AC to DC transformer (220VAC to 12VDC) 7](#_Toc134637040)

[Figure 7: 3.7V 6800mah lithium battery 7](#_Toc134637041)

[Figure 8: 3D printed Robotic Arm 8](#_Toc134637042)

[Figure 9: L298 H-Bridge 8](#_Toc134637043)

[Figure 10: Conveyer Belt with its base 9](#_Toc134637044)

[Figure 11: Universe of discourse of the bottle 10](#_Toc134637045)

[Figure 12: Universe of discourse of the Cup. 10](#_Toc134637046)

[Figure 13: Input membership function of the belt 10](#_Toc134637047)

[Figure 14: Python code for input membership functions 11](#_Toc134637048)

[Figure 15: Python code for the output 11](#_Toc134637049)

[Figure 16:Python code for the rules of the belt controller 12](#_Toc134637050)

[Figure 17: Output to a given input 12](#_Toc134637051)

[Figure 18: Input membership function 13](#_Toc134637052)

[Figure 19: Python code to create input membership function 13](#_Toc134637053)

[Figure 20: Output Membership functions 14](#_Toc134637054)

[Figure 21: Inverse Kinematic 15](#_Toc134637055)

[Figure 22: Python code to create output membership functions 15](#_Toc134637056)

[Figure 23: Sample input testing 15](#_Toc134637057)

# Methodology:

## Design:

To layout the three degree of freedom robot arm with a gripper that is capable of picking up and sorting cups and bottles which are located on a conveyor belt, we accompanied a based technique that concerned numerous key steps. First, we performed a thorough evaluation of the problem and the favored outcome, along with an in depth examine of the conveyor belt, the objects to be picked up, and the overall surroundings. We then evolved a clean and concise specification for the robotic arm, such as its size, weight, and variety of movement, as well as the desired accuracy and precision for object detection and sorting.

Next, we used an existing layout for the robot arm that we got from grabCAD, which include motors, sensors, contemplating factors such as price, reliability, and simplicity of integration.

Once the design was finalized, we proceeded to construct and take a look at the robot arm, iteratively refining and enhancing its capability primarily based on the consequences of our assessments. Finally, we included the arm with the object detection machine and conveyor belt manipulate mechanism, ensuring seamless communique and coordination among all additives. This technique allowed us to design and construct an excellent robot arm that meets the specific necessities of the application efficiently and correctly.

### Robotic arm:

* 3D printed robotic arm
* 3 MG996R servos
* 1 SG90 9g servo
* 2 MPU 6050
* Screws
* 2 Lithium batteries (3.7V)

As shown in the figure, the robotic arm comprises a rotating base link and two moving links, in addition to a gripper that has the ability to open and close. The two MPUs located on the moving links provide the actual angle of each link, which can serve as input for the fuzzy logic controller responsible for the arm's movements.

To power the servo motors, two lithium batteries are utilized, which supply the rated voltage required. Additionally, the servo motors share common ground with the Raspberry Pi.

However, due to a malfunction in one of the MPUs, the arm was ultimately designed to operate with only three degrees of freedom instead of the originally intended four degrees.

### Conveyor belt:

* 12 V DC motor (high torque)
* Metal coupling
* 5 bearings
* 2 pipes
* Rubber
* Screws and bolts
* H-bridge
* AC to DC transformer (220VAC to 12VDC)
* Grease

As shown in the figure above, the conveyor belt is equipped with five bearings, four of which are inserted inside the green pipes. These pipes have a metallic rod inserted through them, which functions as a center axis. Three of these metallic rods will be fixed to the chassis of the belt, while the fourth one will be fixed to the coupling. The coupling, in turn, will be attached to the motor, such that when the motor rotates, the entire belt will rotate.

To power the motor, a transformer is used to supply rated voltage for the h-bridge which also shares common ground with the Raspberry Pi.



Figure :MPU6050



Figure : High Torque 12V DC Motor



Figure : MG996R Servo Motor



Figure : SG90 9g Servo Motor

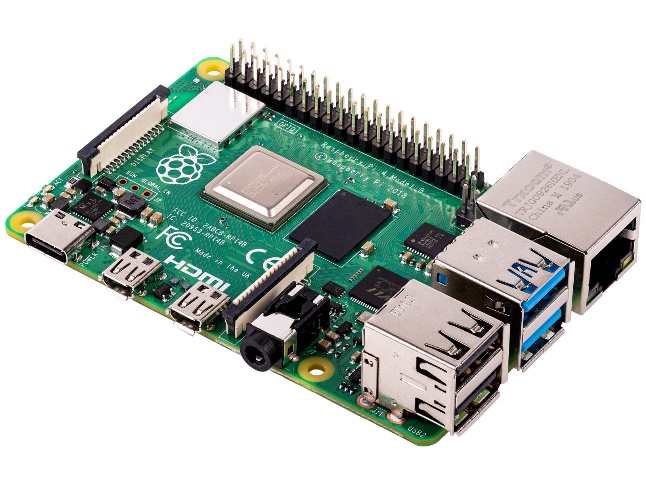


Figure : Raspberry Pi 4



Figure : AC to DC transformer (220VAC to 12VDC)



Figure : 3.7V 6800mah lithium battery



Figure : 3D printed Robotic Arm

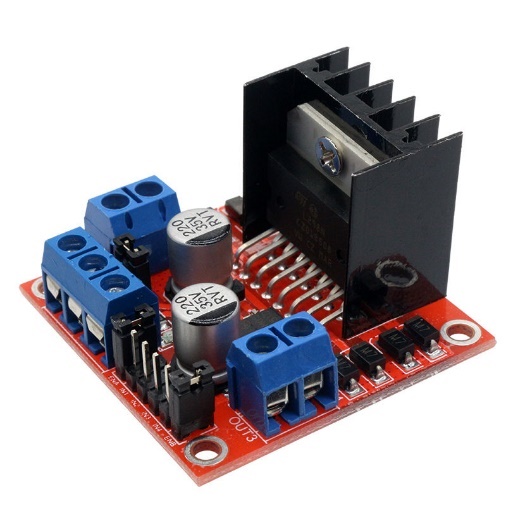


Figure : L298 H-Bridge

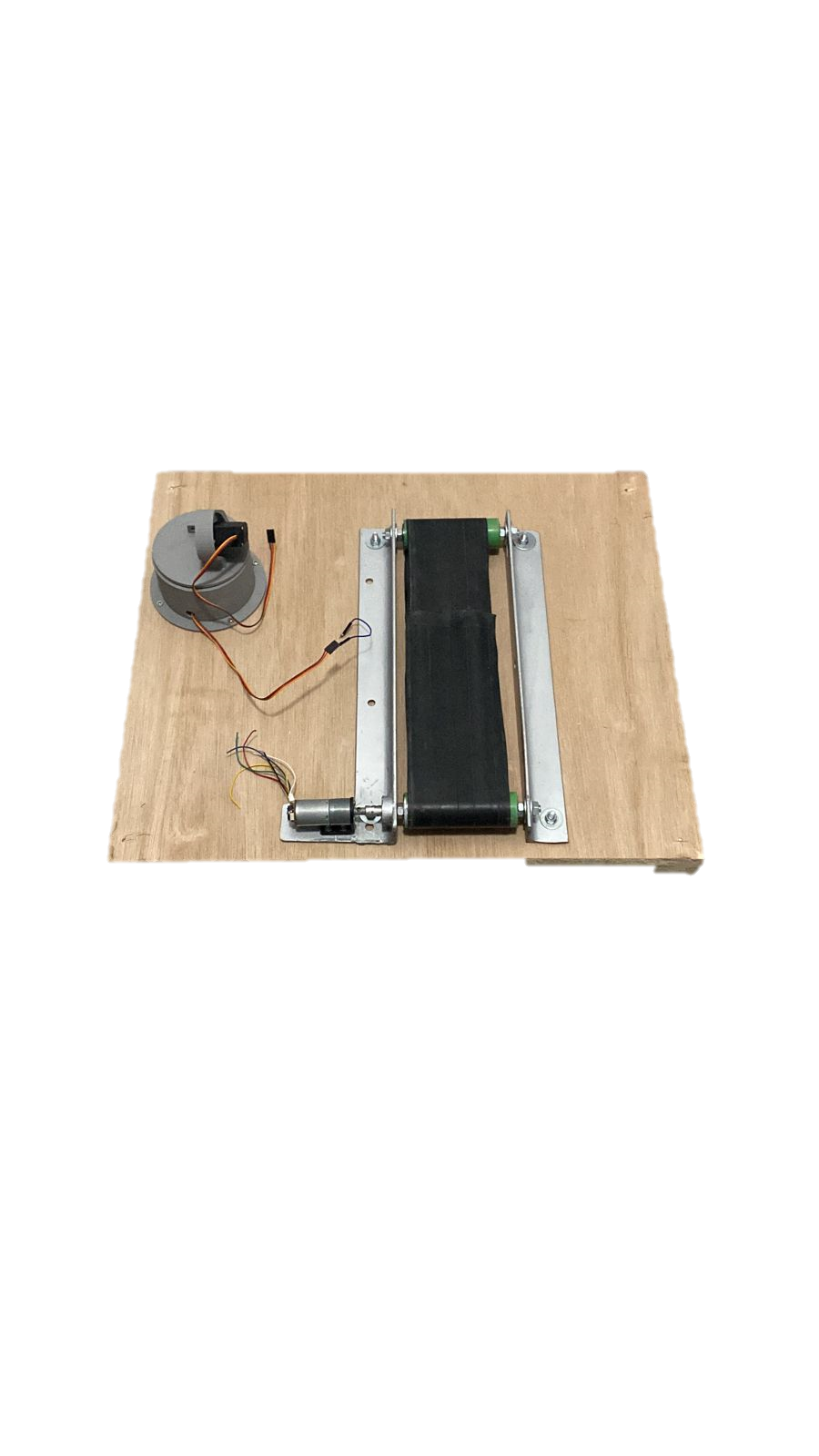


Figure : Conveyer Belt with its base

## Simulation:

We utilized two Fuzzy Logic controllers to control our system. The first controller was used for the belt, and it relied on the centered pixel of the object as feedback. The second controller was used for the robotic arm, and it relied on feedback from the MPU6050 sensor from two different links.

### Belt:

To obtain the universe of discourse, we measured the position of the pixel along the x-axis at different locations on the belt, as well as the desired pixel position. The output of the fuzzy logic is a Pulse Width Modulation (PWM) signal, which is then sent to the motor. Furthermore, we tested the angular velocity for different PWM outputs to determine the desired universe of discourse for the output. All of the codes were implemented using Python and the SKFuzzy library.



Figure : Universe of discourse of the bottle



Figure : Universe of discourse of the Cup.

To take enough data about the x-coordinates that will be faced in our application, we turned on the motor with the bottle on the belt so that the model can detect the bottle on different positions until reaching the desired position. Then we printed the x-coordinates for all the positions that were detected by the model as shown above. The same was done to know the universe of discourse for the cup. From the data shown in the figure above we chose that the universe of discourse must be between 900 and 580 (which is the desired position). Also, from the increments we got we chose the membership functions for the input as shown in fig.13:

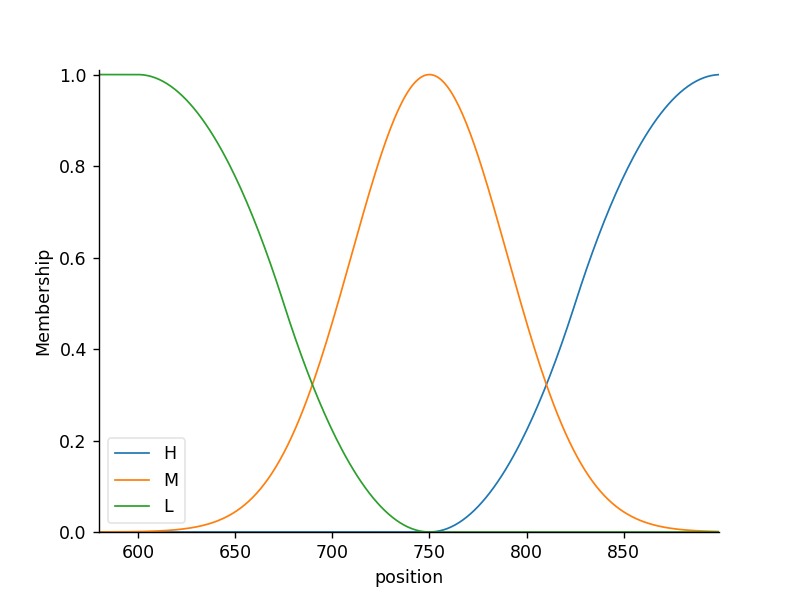


Figure : Input membership function of the belt

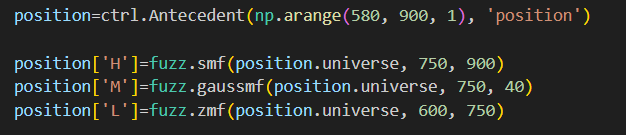


Figure : Python code for input membership functions

The universe of discourse is [580,900].

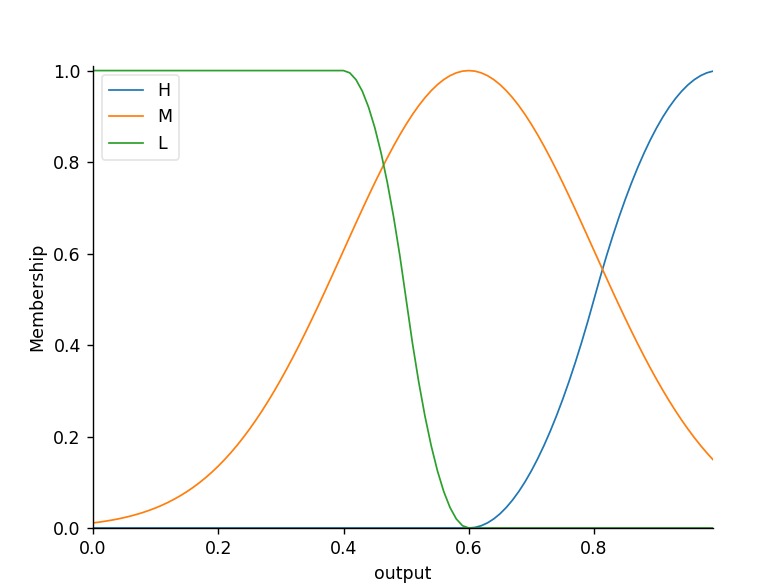
Our desired pixel is 580.

Low is from 580 to 750.

Medium is 600 to 900.

High is from 750 to 900.

All parameters have been chosen experimentally.



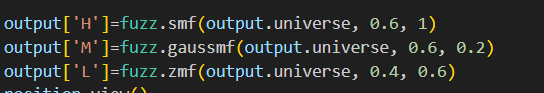


Figure : Python code for the output

Universe of discourse is [0 1].

Low is from 0 to 0.6.

Medium is 0 to 1.

High is from 0.6 to 1.

Also, parameters have been chosen experimentally. The experiments were testing the speed of the belt when subjected to multiple PWM signals.

A picture containing font, text, screenshot, graphics

Description automatically generated

Figure :Python code for the rules of the belt controller

The rules were as follow:

If the position is high then the output is high.

If the position is medium then the output is medium.

If the position is low then the output is low.

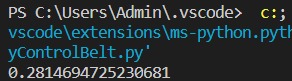


Figure : Output to a given input

For an input of 650 which means that our center of our object x-axis coordinate is equal to 650 the output to be given to the motor is 0.28.

### Robotic arm:

The assumed input universe of discourse ranges from -90 to 90 degrees, since the servo motor rotates from 0 to 180 degrees. As a result, if the error were to be greater than 90 or less than -90 degrees, it would indicate a malfunction in the servo motor. The output universe of discourse is set between -10 to 10 to ensure that the process runs smoothly, without any sudden or aggressive changes.

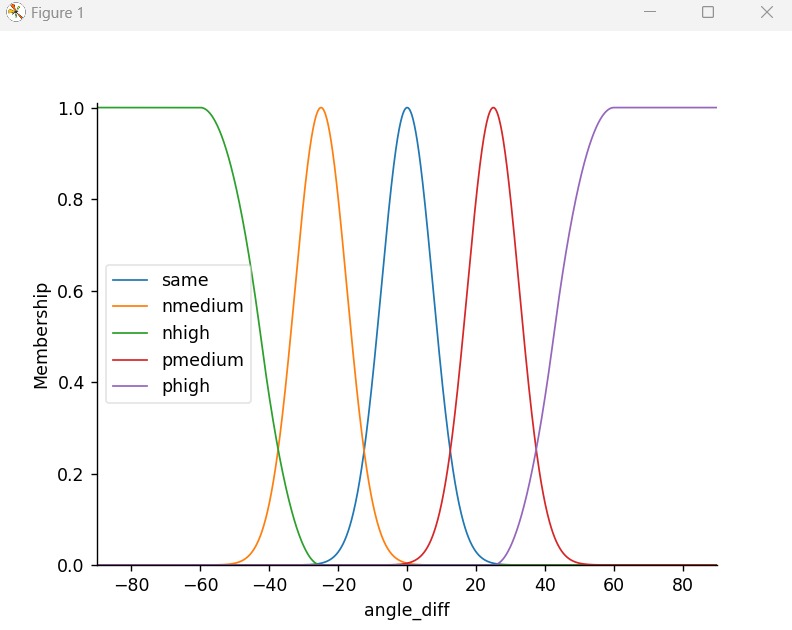


Figure : Input membership function

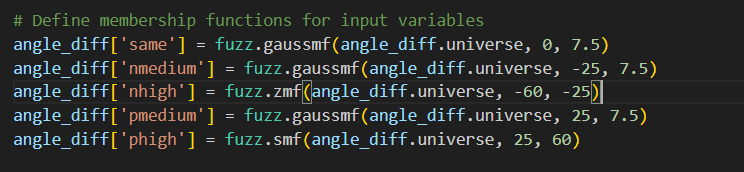


Figure : Python code to create input membership function

The input provided is the difference between the desired angle and the current angle of the links. The angle difference can fall within different ranges. It can be the same, ranging from -25 to 25. It can also be negative and medium, ranging from -50 to 0. Alternatively, it can be negative and high, ranging from -90 to -25. Additionally, it can be positive and medium, ranging from 0 to 50, and finally, positive, and high, ranging from 25 to 90.

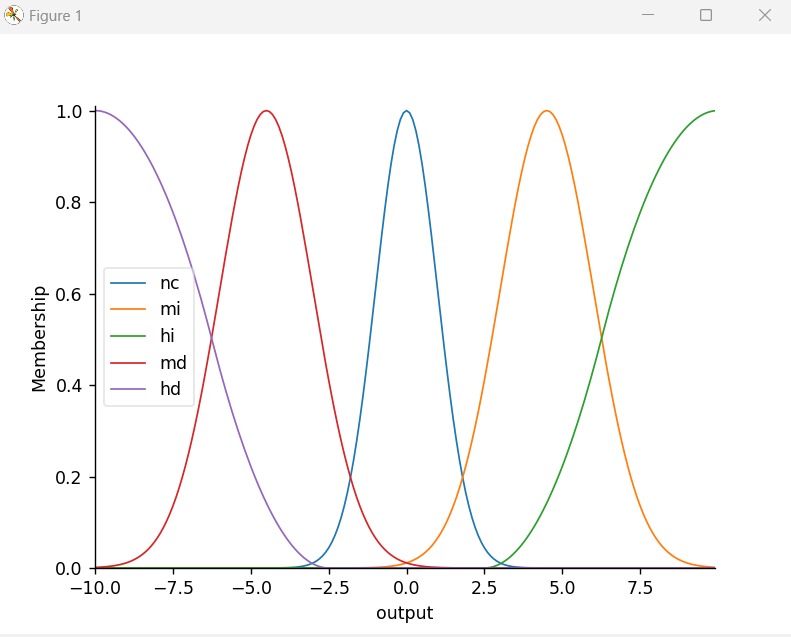
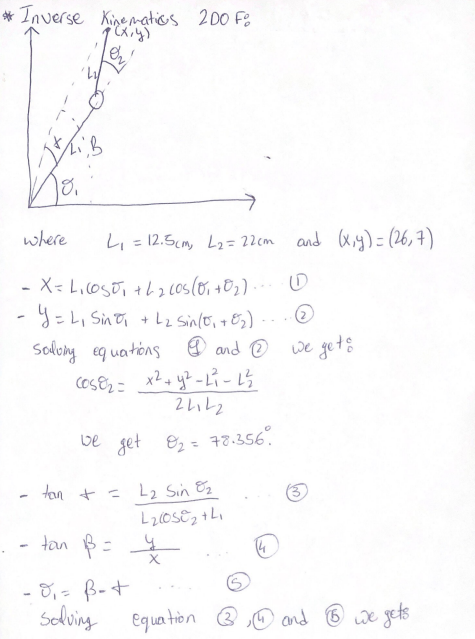


Figure : Output Membership functions

The resulting output is the angle that needs to be added to the current servo motor angle. This angle can also indicate no change, ranging between -3 to 3 degrees. It can also indicate a medium increase, ranging from 0 to 8 degrees, and a high increase, ranging from 2.5 to 10 degrees. Additionally, it can indicate a medium decrease, ranging from -10 to 0 degrees, and finally, a high decrease, ranging from -10 to -2.5 degrees.



A picture containing text, handwriting, font, calligraphy

Description automatically generated

Figure : Inverse Kinematic

To determine the desired angle output for each link, we used inverse kinematics after measuring the x and y coordinates of our object with respect to the center of our robotic arm:

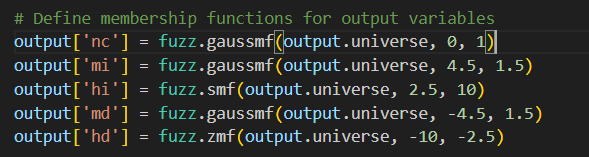


Figure : Python code to create output membership functions

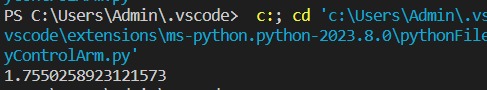


Figure : Sample input testing

An example output was obtained when the input value was set to 10, resulting in an output of 1.755.

## Implementation:

### Process of sorting:

Our object detection code relied on a pretrained YOLOv4 model running on a laptop with an external webcam. To communicate with the Raspberry Pi, we established a socket connection, where the laptop acted as the server, transmitting information to the client (Raspberry Pi). The client was designed to wait for a connection before executing the code. Mainly the code on the laptop will take the video from the webcam, use the computer vision model to detect at each frame if there is bottle or cup. Moreover, the x-coordinate position of the object will be fed to the fuzzy logic controller, and the controller will provide the needed PWM signal for the motor. Then this output will be transmitted as a string to the raspberry pi code through a socket, the code on the pi will decode it and feed it to the motor.

Once the object detection and fuzzy logic control of the belt processing were complete, the laptop would send a required PWM signal to the motor. The fuzzy logic controller would then execute again on the laptop to obtain another PWM signal, appropriate for the new position of the object. This process would repeat until convergence was reached, at which point the roles would reverse, with the Raspberry Pi becoming the server and the laptop becoming the client.

After the belt fuzzy controller converged, we used a fuzzy logic controller to set the robotic arm in the desired position, grab the object, and transport it to its corresponding location on the left or right of the arm. Once the object was sorted, the Raspberry Pi would send a signal to the laptop via sockets indicating that the process was complete, triggering the object detection code and fuzzy logic controller of the belt on the laptop to repeat the process.

## Problems in the testing phase:

1. Model Training: First, we decided to train our own model that supports and differentiates between two different well-defined objects. So, we started with a dataset of 300 images and we used roboflow to annotate our images and got our XML files. Now for the training we trained using the mobilenet SSD V2 (which has very fast inferencing time and high accuracy). After the training had finished, we converted our model to a tflite model and deployed it on the raspberry pi. When it was deployed the model was acting weird and was detecting everything and labeling them as a bottle. Second, we thought that we needed a bigger dataset so we took 500 more pictures to have a total of 800 images to train our model. In addition to the new pictures, we augmented the data to finally have a total of 2100 images to train our model. However, we still had the same error. That is why we decided to use a pretrained YOLOV4 model that we downloaded online. This model can differentiate between 73 classes but we only used two classes (bottle and cup).
2. MPU calibration: When the MPUs were tested they were giving inaccurate data so we needed to apply the Kalman Filter algorithm which stabilized the outputs of the MPUs. The code for the Kalman filter algorithm was downloaded from a GitHub repository named Kalman.
3. 3 DOF: The initial design of the robot was a four degree of freedom mechanism, however one of the three MPUs was defective, so we fixed one of the Links.
4. Resistors: In order to have one power supply for the machine, we tried to use 4.7-ohm resistors to divide the voltage from 12V to 5V to feed the servo motors. However, the resistors did not withstand the current absorbed by the servos. To solve this problem, we were obligated to use two 3.7V lithium batteries to supply the servo motors alone.
5. Servo Motors: The MG996R servo motors have a problem in holding their position which creates some oscillations in their behavior. Also, when testing the gripper its servo motor malfunctioned suddenly, so we used another gripper that was available that uses the SG90 servo motor.

# Results and Limitations:

## Limitations:

1. The robotic arm can be implemented with more stable servo motors which can increase the accuracy of the machine.
2. Our model operates properly in certain lighting conditions, a better model can be implemented which will make the machine more reliable.
3. Our machine is trained to sort only two objects, which are the bottle and the cup.
4. Camera calibration techniques can be used so that the model can operate properly regardless of its position.

## Results:

We are pleased to present the results of our project which exceeded all expectations and proved to be a great success. Our team’s unwavering commitment that helped us overcome all the challenges that we encountered through our journey in this project, meticulous planning and relentless pursuit of excellence have allowed us to achieve incredible results. By employing innovative techniques and using cutting-edge technology we were able to make the automated sorting machine that was able to continuously sort a bottle and a cup using computer vision. All fuzzy controllers were working with high efficiency and accuracy and the connection between the laptop and the raspberry pi was very effective and working very smoothly.

Project git hub repository:

https://github.com/RamiHjeij/Autonomous-Sorting-Machine.git

Video: <https://drive.google.com/file/d/1pVdlkhVqWchsBrMoY4FO01Fo6XS_HtjN/view?usp=sharing>

# References:

<https://github.com/rocheparadox/Kalman-Filter-Python-for-mpu6050.git>