ALGORITHM

Multi Robot-shape formation refers to the process of coordinating a group of autonomous robots to collectively arrange themselves in a desired geometric shape or formation. This problem has gained significant attention in the field of multi-robot systems and has applications in various domains, such as robotic swarms, cooperative manipulation, and surveillance.

To achieve shape formation, it is essential to develop efficient algorithms that allow the robots to communicate, coordinate their movements, and self-organise in a decentralised manner. These algorithms typically leverage local sensing, local communication, and distributed decision-making to enable the robots to achieve the desired shape while adapting to changes in the environment or the robot group.

The Algorithm of Multi Robot-shape formation is mainly divided into three parts:

1. Real-Time location Tracking of Robot
2. Setting set points, according to shape
3. Communication between Robot and Camera

1. Real-Time location Tracking of Robot

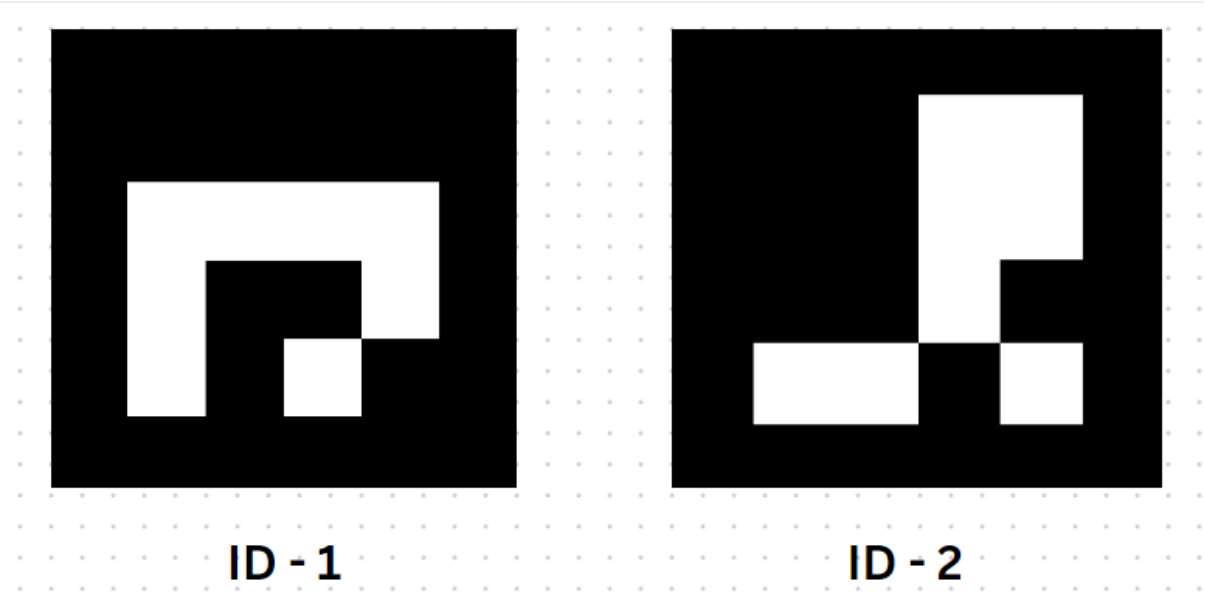
Real-time location tracking of a robot refers to continuously determining and updating the robot's position and orientation in the environment as it moves.There are various approaches to achieve real-time robot location tracking, But we are using “ArUco marker-based localization in OpenCV”, Which provides a reliable, accurate, and efficient solution for real-time robot localization, with the added benefits of flexibility, community support, and integration with other computer vision functionalities.

ArUco Marker

ArUco markers are square-shaped markers with black and white patterns that are used in computer vision applications, particularly for marker-based tracking and localization. These markers are designed to be easily detectable and identifiable by computer vision algorithms. ArUco stands for "Augmented Reality University of Cordoba."

Each ArUco marker has a unique ID encoded within its binary pattern. The ID serves as a marker identifier, allowing for individual marker detection and identification.

The binary pattern of an ArUco marker contains a grid-like arrangement of black and white squares. The presence or absence of black squares in specific locations within the pattern represents the ID. By analysing the pattern and decoding the binary information, computer vision algorithms can determine the unique ID associated with each marker.



ArUco markers are typically printed and placed in the environment or attached to objects to serve as reference points. By detecting and tracking these markers in an image or video stream, their poses (position and orientation) can be estimated, allowing for tasks such as object tracking, augmented reality, or camera calibration.

The OpenCV library provides built-in functions for ArUco marker detection, identification, and pose estimation, making it convenient to integrate ArUco marker-based functionalities into computer vision applications.

Aruco Marker-based localization in Open CV

Marker Generation:

ArUco markers are generated using the ArUco library in OpenCV. The library provides functions to create markers with unique binary patterns based on a specified dictionary and marker size. The dictionary contains a set of predefined markers with distinct patterns. You can choose the dictionary type and marker size based on your requirements.

Camera Calibration:

Camera calibration is a crucial step to obtain accurate localization results. It involves determining the intrinsic and extrinsic parameters of the camera.
OpenCV provides functions for camera calibration, such as cv::calibrateCamera, which uses a calibration pattern with known 3D positions to estimate the camera's intrinsic parameters (focal length, principal point, and distortion coefficients) and extrinsic parameters (rotation and translation).

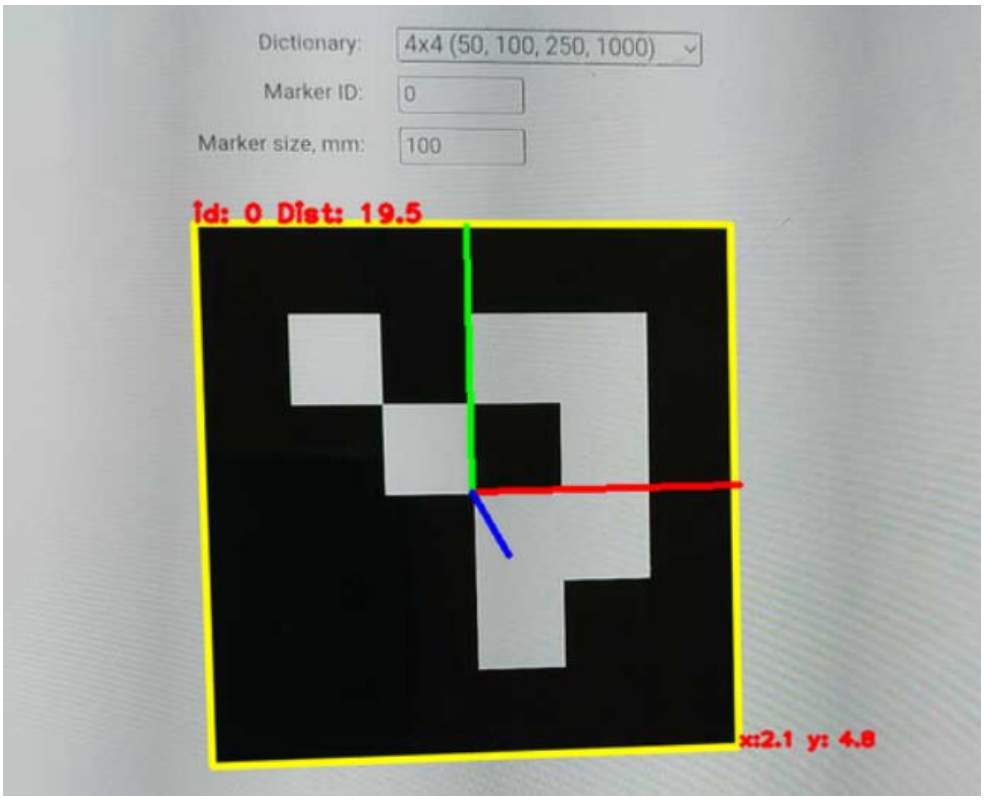
Camera calibration ensures accurate mapping between the camera's image coordinates and the 3D world coordinates.

Image Acquisition:

Real-time video frames are acquired from a camera using OpenCV's video capture functionalities. These frames will be processed to detect and track the ArUco markers.

Marker Detection:

OpenCV provides functions like cv::aruco::detectMarkers for marker detection. This function takes the acquired video frames and the predefined dictionary as input.
Marker detection algorithms analyse the frames to identify and locate the ArUco markers based on their unique binary patterns. The function returns the corners of the detected markers.



Pose Estimation:

Once the markers are detected, their pose (position and orientation) is estimated in the camera coordinate system. OpenCV's cv::aruco::estimatePoseSingleMarkers function estimates the pose of individual markers. It takes the detected marker corners, camera calibration data, and marker size as input. The function calculates the translation vector (representing the marker's position) and the rotation matrix (representing the marker's orientation) for each detected marker.

Coordinate Transformation:

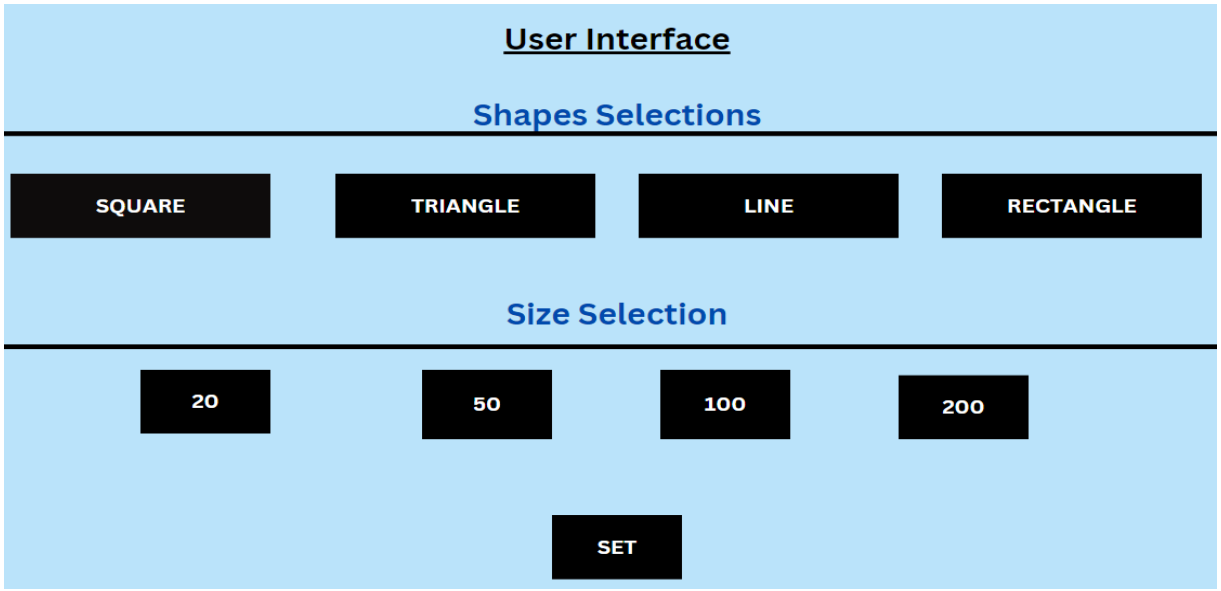
To localise the camera or robot, coordinate transformations are applied to the marker poses. The transformations convert the marker poses from the camera coordinate system to a desired reference coordinate system.
If the positions of the markers in the reference coordinate system are known, techniques such as triangulation or averaging the marker positions can be used to calculate the camera or robot's position.
The transformation from camera coordinates to the reference coordinate system can be computed using the extrinsic parameters obtained during camera calibration.

Real-Time Tracking and Updates:

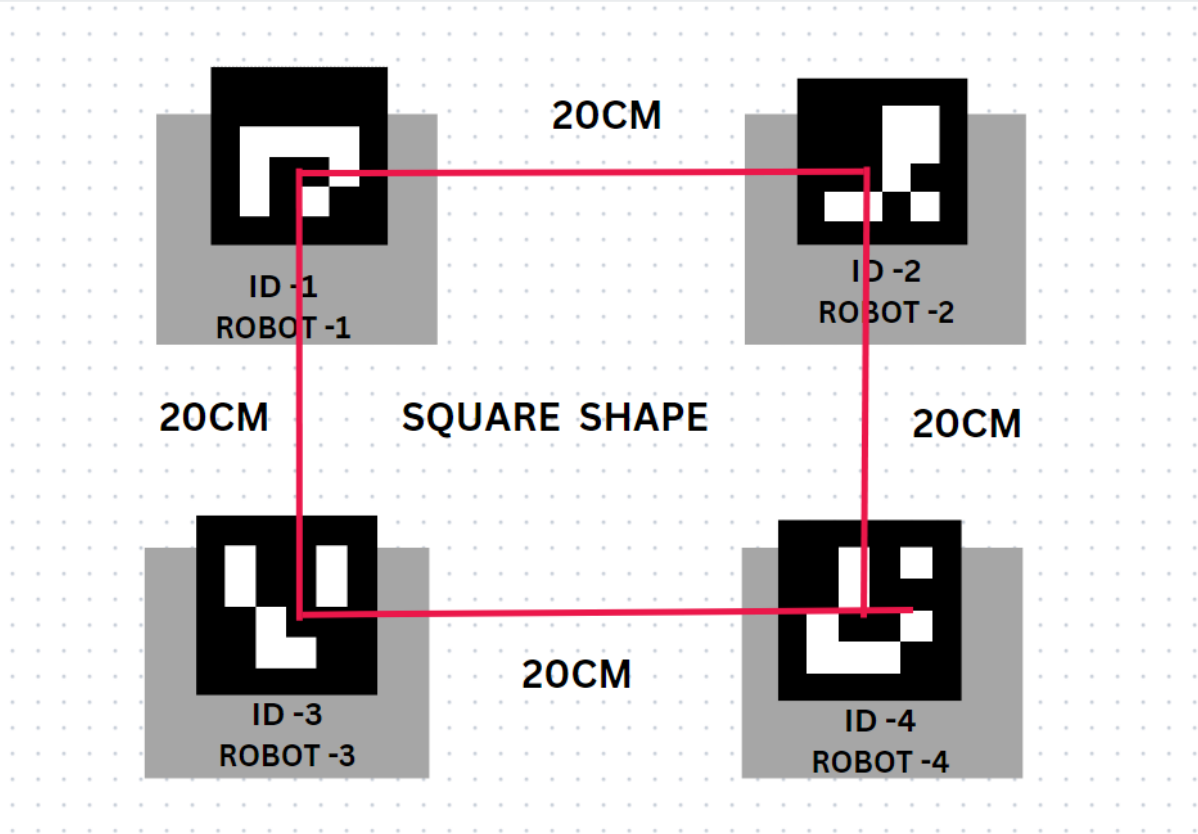
ArUco marker-based localization is performed continuously on the video stream to track the camera or robot's location in real-time. As new frames are acquired, the markers are detected, their poses are estimated, and the camera or robot's position and orientation are updated accordingly.This real-time tracking allows for dynamic localization, facilitating tasks such as navigation, mapping, or augmented reality. ArUco marker-based localization in OpenCV provides an accurate and efficient method for real-time camera or robot localization.

2. **Setting set points, according to shape**

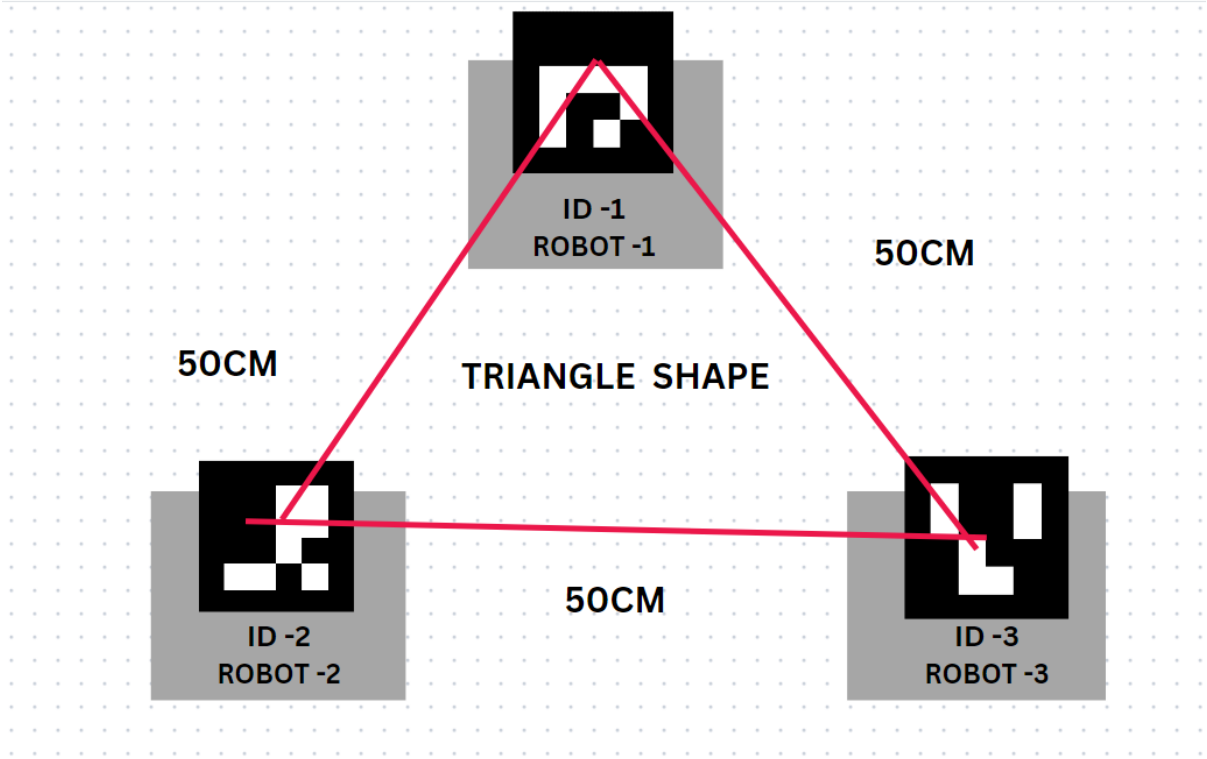
Graphical-User interface is formed to select the shapes and size of the shapes the user wants to form . According to the shapes and sizes set by the users , the set point which is already defined is visible on the screen and its coordinates are assigned to the code and by coorecting the distance and angle with that set point, the robot gets the distance it has to travel to form the selected shape.



User-interface consists of a rectangle, triangle,square and line with different size, when the user selects the shape and size . For example , if the shape is square and size is 50 ,then the robot forms a square shape with 20 cm length.

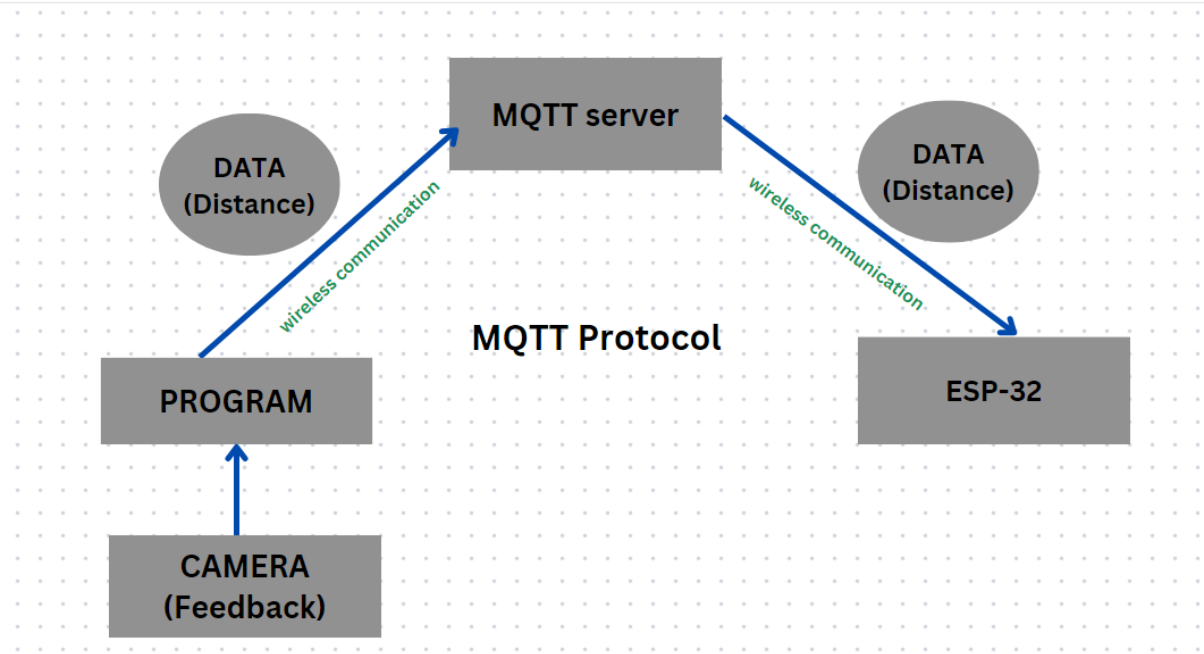


If Triangle is selected then the first three bot form Triangular pattern with selected size as shown in below figure, the fourth robot is not included in this shape formation , it is parked outside the shape at decided set point.



3. Communication between Robot and Camera

Communication plays a very important role in error correction in this system , when the distance is calculated by the program which it gets from the camera's feedback is transmitted to esp-32 board. So,it moves according to the distance .This Transmission takes place by using the MQTT protocol.



MQTT (Message Queuing Telemetry Transport) is a lightweight publish-subscribe messaging protocol designed for constrained devices and low-bandwidth, high-latency or unreliable networks. It is commonly used in Internet of Things (IoT) applications to enable communication between devices and applications.

Some of the key aspects of the MQTT protocol:

Publish-Subscribe Model:

MQTT follows a publish-subscribe model, where devices can publish messages to topics, and other devices or applications can subscribe to those topics to receive the messages. Topics are hierarchical and can be organised using a topic structure, such as "home/bedroom/temperature".

Broker:

MQTT requires a message broker, which acts as an intermediary between publishers and subscribers. The broker receives messages published to topics and distributes them to the relevant subscribers. It also maintains information about the subscriptions and handles the routing of messages.

Quality of Service (QoS): MQTT supports three levels of QoS to ensure message delivery reliability.

QoS 0 (At most once): Messages are delivered once but may get lost or duplicated.

QoS 1 (At least once): Messages are guaranteed to be delivered at least once, but there may be duplicates.
QoS 2 (Exactly once): Messages are guaranteed to be delivered exactly once, ensuring no duplicates.

Lightweight:

MQTT is designed to be lightweight and efficient, making it suitable for resource-constrained devices with limited processing power, memory, and bandwidth.

Connection-oriented: MQTT is a connection-oriented protocol, meaning that clients establish a persistent TCP/IP connection with the broker. This allows for efficient communication with low overhead and reduces the need for frequent connection setup and teardown.

Security:

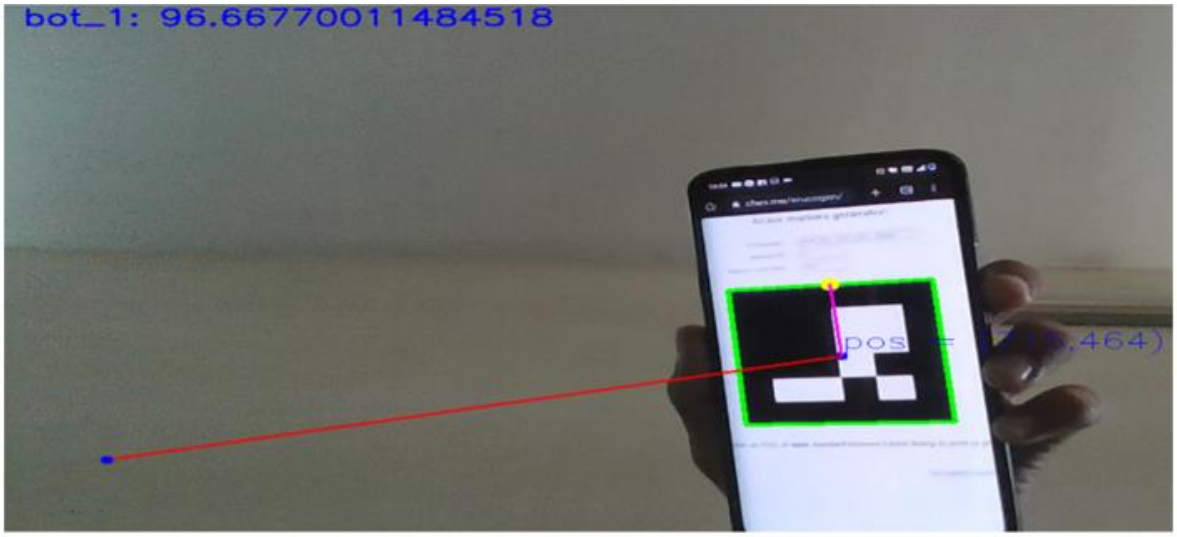
MQTT does not provide built-in security mechanisms, but it can be used over secure transport protocols such as TLS/SSL to establish an encrypted connection between clients and brokers.

Retained Messages:

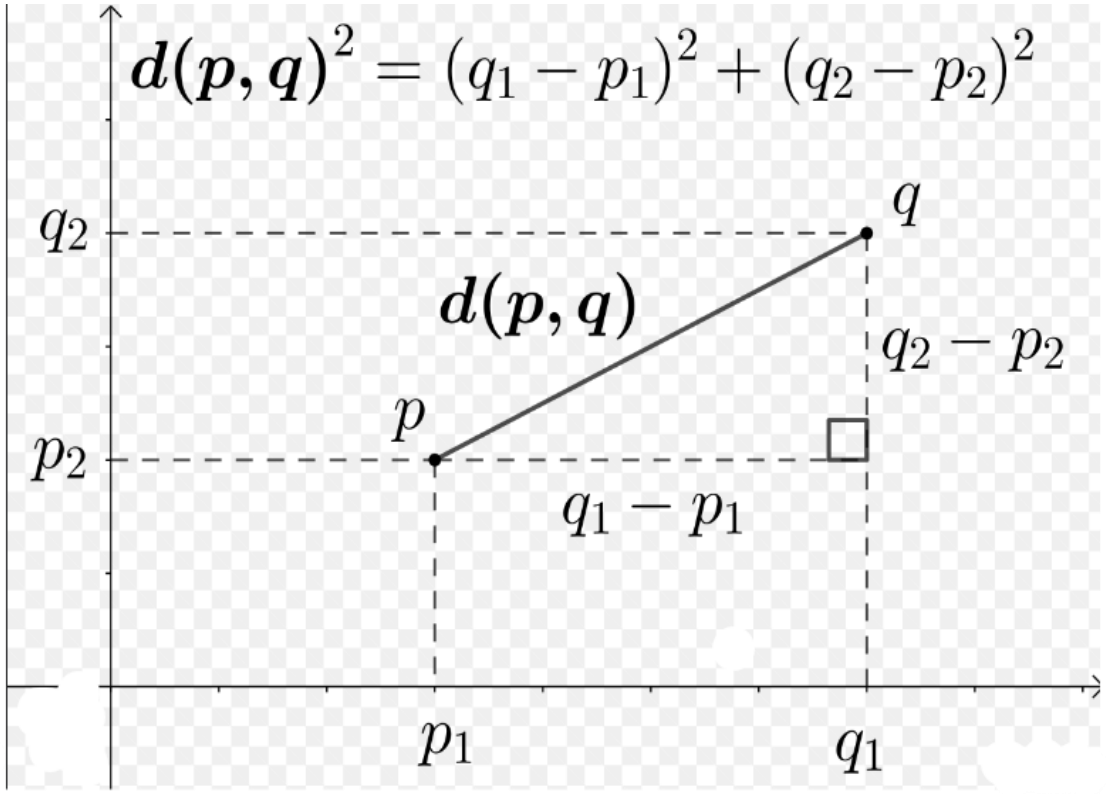
MQTT allows publishers to mark messages as "retained," meaning that the last message published on a topic will be retained by the broker and sent to any new subscribers that join the topic. This is useful for transmitting status updates or configuration information.

TRAJECTORY PLANNING

As shape and size is selected from the Graphical-user interface, the set point is already fixed according to the selection of size and shape. The Camera detects the ArUco code, as the ArUco code is detected ,it shows the angle between midpoint and the endpoint on the ArUco as shown in figure.



As the angle is detected it is transmitted to esp-32 with Mqtt protocol and if the angle is greater or less than zero degree then it rotates the robot in 360 degree unless the angle becomes zero degree. As the angle is zero then it measures the distance between two coordinates.



As the distance decreases , the robot is also getting nearer to the set point , as its reaches near the set point , the motor stops rotating forward and the robot reaches its set point . All the robot one by one reaches to its set point and forms the selected shape with selected size.