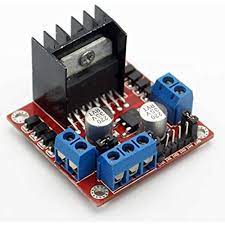
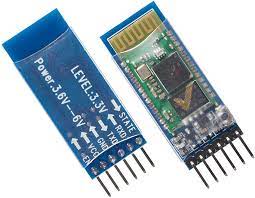
**Methodology**

**a) Resources Utilized (HARDWARE AND SOFTWARE)**

**Hardware Components**

* Arduino UNO
* ESP32-Cam
* L298N Motor Driver
* HC-05 Bluetooth Module
* DC Motors
* Ultrasonic sensor
* LED
* Buzzer

  **Software Components:**

* Arduino IDE
* VS code
* Flask
* OpenCV
* YoloV3



The Arduino Integrated Development Environment (IDE) is a free, user-friendly software platform for writing, compiling, and uploading programs to Arduino boards. It's a graphical user interface (GUI) that provides a range of tools and features to simplify the development process.



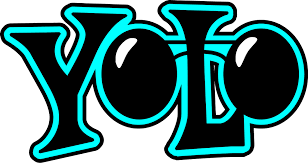
VS Code is a free, open-source code editor developed by Microsoft. It is a lightweight, extensible, and customizable editor that supports a wide range of programming languages, including Python, JavaScript, TypeScript, and more. VS Code offers features such as code completion, debugging, and version control integration, making it a popular choice among developers.

Flask is a Python-based web application framework that allows developers to build web applications quickly and efficiently. It's a microframework, which means it's lightweight and doesn't require specific tools or libraries. Flask is a popular choice for development teams working on small to medium-sized projects.



OpenCV (Open-Source Computer Vision Library) is a software library that's used for computer vision and machine learning. It's a standard tool for image processing and computer vision, and is used by many companies to develop applications for these fields.

You Only Look Once or YOLO is an algorithm capable of detecting objects at first glance, performing detection and classification simultaneously. Find out all you need to know about this revolutionary approach to AI and Computer Vision! Object detection is one of the mainstays of computer vision.



**b) Experiment Details (Procedure):**

**Setup and Connections:**

**For Train:**

**Pin Configuration:**

Pin 13: Left motors forward

Pin 12: Left motors reverse

Pin 11: Right motors forward

Pin 10: Right motors reverse

Pin 9: motor Stop pin

**Communication Setup:**

Serial communication is initialized at 9600 baud rates.

**For ESP32-CAM:**

**Communication Setup:**

Begins serial communication at 115200 baud rates.

**Pin Configuration:**

Sets TrigPin as OUTPUT (for ultrasonic sensor)

Sets EchoPin as INPUT (for ultrasonic sensor)

Sets buzz pin as OUTPUT (for buzzer)

Sets MotorStopPin as OUTPUT (to control motor stopping)

**Camera Setup:**

Configures the ESP32-CAM module using the esp32cam library

Sets camera pins for the AI-Thinker ESP32-CAM board

Sets the initial resolution to high resolution (hiRes)

Initializes the camera and prints whether it was successful

**Wi-Fi Connection:**

* Sets Wi-Fi to station mode.
* Connects to the specified Wi-Fi network
* Waits in a loop until the connection is established
* Prints the local IP address of the ESP32 once connected

**Web Server Routes:**

Sets up HTTP routes for the web server:

"/cam-lo.jpg" for low-resolution images

"/cam-hi.jpg" for high-resolution images

"/cam-mid.jpg" for medium-resolution images

"/update\_detection" for receiving detection results

**Start Web Server:**

Begins the web server to listen for incoming requests

***Functionality of Arduino Uno:***

**Movement of the train:**

**Move Backward ('B'):** When the command 'B' is received, the robot activates both left and right motors in reverse. This causes the robot to move backward.

**Turn Left ('L'):** Upon receiving the command 'L', the robot stops the left motors while activating the right motors to move forward. This results in the robot turning left.

**Turn Right ('R'):** When the command 'R' is received, the robot stops the right motors and activates the left motors to move forward. This action causes the robot to turn right.

**Stop ('S') or Motorpin (High):** If the command 'S' is received or if the button connected to a specific pin is pressed, the robot stops all motors, effectively bringing it to a halt.

**Default Movement:** If none of the recognized commands are received, the robot defaults to moving forward. In this case, both left and right motors are activated to move the robot ahead.

**Functionality of ESP32 cam:**

Handling HTTP Requests:

*/*Update\_detection: Receives JSON data indicating whether living or non-living objects have been detected. It checks the distance to the detected object and activates the buzzer or stops the motors based on the conditions defined.

/cam-lo.jpg, /cam-mid.jpg, /cam-hi.jpg: Routes for capturing images at different resolutions. Each route changes the camera resolution before capturing an image.

**Distance Measurement:**

The readUltrasonicDistance function measures the distance using the ultrasonic sensor and returns it in centimetres.

**Image Capture:**

The serveJpgWithDistance function captures an image and sends it back to the client along with the distance measured by the ultrasonic sensor in the HTTP headers.

**Functionality of Python:**

Loading YOLO Model: The application loads the YOLO model's weights and configuration files. It also retrieves the names of the layers in the model, which are necessary for processing the output of the model.

Class Names: The application reads a list of class names (like 'person', 'car', etc.) that the YOLO model can detect from a file. This helps in labelling detected objects.

Video Streaming: The application continuously fetches frames from the ESP32-CAM. It processes these frames to detect objects using the YOLO model.

Detection Logic: For each frame, the application processes the image to identify objects. It uses confidence scores to filter out detections that are not reliable. Detected objects are then labelled and drawn on the frame.

Distance Measurement: The application retrieves distance information from the ESP32-CAM's response headers, which could be used for various applications (like measuring how far an object is from the camera).

Updating ESP32-CAM: Based on the detections (whether living or non-living objects are found), the application sends a POST request to the ESP32-CAM to update its status.

Web Interface

HTML Structure: The application serves a simple HTML page that displays the video feed from the ESP32-CAM alongside a list of detected objects and their confidence levels.

JavaScript for Updates: The web page uses JavaScript to periodically fetch the latest detection results and distance from the server, ensuring that the displayed information is up to date.

Styling: Basic CSS is used to style the layout, making the video feed and detection information visually appealing and organized.

Execution

When the application is run, it starts a web server that listens for incoming connections. Users can access the web interface through a browser, where they can view the live video feed and see the detected objects along with their confidence scores.