B.Tech IoT Programming Project Report

On

Automated Parking System Using YOLOv8

Team Members:

- Pratik Gond (123103006) 123103006@nitkkr.ac.in
- Saransh Gupta (123103002) 123103002@nitkkr.ac.in
- Kush Bohare(123103005) 123103005@nitkkr.ac.in
- Shubham Mahile(123103007) 123103007@nitkkr.ac.in

Abstract

In our Project “Automated Parking System using YOLOv8,” we have focused on improving the existing parking system's reliability and functionality by implementing IoT wherever necessary and feasible to minimize human intervention.

First of all, we are using the YOLO (You Only Look Once) algorithm, a deep learning object detection model, which detects that the object coming at the parking area is a vehicle or not. If it is not a vehicle, no further processes are conducted. But if it is a vehicle, the algorithm captures the number plate using optical character recognition (OCR) and stores the vehicle number along with the entry time into the cloud-based database. This data is necessary for tracking the vehicle within the system and is handled via IoT communication protocols, such as MQTT or

HTTP.

After YOLO sends this data to the database, the further process is triggered. The Node MCU recieves instruction of (1/0) from cloud to check if any parking slot is empty. There are numerous parking slots in the parking area, and each parking slot has an ultrasonic sensor installed. These sensors check if any slot is available and send the data regarding available slots to the Node MCU installed at the parking gates. It uses this data to ensure if it has to open the gates or not.

In addition, we have used an LCD (I2C) near the entry gates of the parking area, which displays the available parking slot number or a message that no space is available, depending on the current status of the slots. This LCD is also stimulated by the same Node MCU.

When a vehicle goes to exit the parking area, the cameras at the exit gate once again work and the YOLO algorithm is used to capture the number plate and mark the exit time of vehicle. When the exit time is recorded, the system opens the gates by following the same process, allowing the vehicle to exit smoothly. All this data is also stored in the database, so that this data can be used for further improvements.

The YOLO Vehicle and Pedestrian Detection used by our project has and solid accuracy of 94% with an even better 95% precision. The Recall percentage and F1-score goes up even higher to 95.7% and 95.34% respectively.The mAP@0.50 (Mean Average Precision at IoU threshold of 0.50) is an excellent 94.61% with the overall IoU (Intersection over Union) being 0.83.

The Ultrasonic sensors to be used in parking slots have an accuracy upto 98% for closer objects which is of our case.

Introduction

The growing number of vehicles in cities has made parking into a major issue. Traditional parking systems aren’t really efficient and depend too much on manual operations. This lead to traffic jams, drivers wasting time looking for parking spaces, and also causes more pollution. Our project, “Automated Parking System using YOLOv8,” is trying to solve these problems by developing a smart, automated parking system that requires less human intervention.

A study [1] shows that on an average people in delhi spent 20 minutes of their everyday for finding parking spaces with 38% of them stating even more time than this. But a simple thing like parking a car must not take more than 5 minutes normally. Thus by Implementing this project we try to save precious time of users.

Drivers keep roaming around to find parking spaces thus causing unnecessary fuel combustion and Noise generation. The same study shows that more than 22% of people don’t go for traditional manual parkings because they dont really know if they would have proper parking space .So with our idea , drivers may simply leave if no space is empty , and reach to their destined slot if its available.

Many a times due to overparking , which happens due to human negligence , the vehicles parked inside are not able to exit . This problem is just very frustrating because then we have to wait even to just leave the area .Our system ensures that no entry is made if all slots are full. Thus providing a smoother experience to our users.

A survey [2] shows that 86% of the total 155 respondents have difficulty to find car park at multilevel parking. It highlights the user pain in finding the parking lot. It should take into account that most of the drivers have problem to find car park and thus there is urgent demand to develop a product which can fulfill their needs.

Our project also aims on minimizing human intervention in the parking procedure. At many places there are at least two people to manage a small parking area, but we believe that some simple works like checking empty slots and opening and closing the gates must not be requiring human labor, simple innovation and implementation can make this process much faster and simple.

Some unique things that we have tried to implement :

1. We have used ultrasonic sensors in our project rather than infrared sensors or photodiodes that are used in most of the other projects. Ultrasonic sensors are more accurate than IR sensors and does not depend on the lighting conditions of the area.
2. We have used vehicle detection with YOLO to ensure that gates are to be opened only if a vehicle has arrived . Also the further process is triggered only if a vehicle is at the entry gate. Rest of the time , our system is not working , thus saving considerable amount of energy.

3. The parking timer starts as soon as the vehicle enters the parking area and stops only when it has exited , thus forcing drivers to not waste unnecessary time in the parking area.

Related works

A significant amount of work has been put into Automatic Parking Management field by various people across the globe. Some use sensors for detection and some are at visual detection of available spaces. ML models are required for visual detection of empty spaces.

The project made by **Sriram et al. [3]** have used a deep ML model R-CNN for the detection of the empty slots using a camera that is installed in the parking area. It uses geo location to mark the empty space if available. They had a success rate of around 93%.

In **Padmasiri et al. (2020) [4]**, a scalable software architecture solution is proposed which enables reliable implementation of an end-to-end automated vehicle parking occupancy detection system. The vehicle detection is performed in real-time with two-stage detectors Faster R-CNN and RetinaNet.

The work done by **Agrawal Et al. (2021) [5]** is using IR sensors at each slot to detect the empty slots available and uses Wi-Fi modules to connect their components to open the gates . They update their data 24x7 onto their web based application.Similar application is done by **Patro Et al. 2020 [6]** and they also have used IR sensors but their work also include separate parking for 2 wheelers and 4 wheelers.

Kanan et al. (2020) [7] uses an different model that uses a VTD , the vehicle transceiver device (VTD) consists of processor, a power management unit, a radio transceiver, an RF wake-up sensor, and global navigation satellite system (GNSS). This VTD remains idle until the RF wake-up sensor doesn't give a wake up call.

Thakre et al. (2020) [8] tends to use an wireless transmission technology RFID (Radio frequency identification) which has users already registered into database and has their accounts made , from which fees is automatically deducted based on their usage of parking service.

Another interesting implementation is done by **Patil et al. (2023) [9]** . They have used CNN algorithm directly with the CCTV camera of the parking that is activated when the driver uses their application and informs that he wants to use that parking area , then the slots are checked and information is updated in the App.

Wang et al. (2020)[10] has proposed an AGV (Automated guided vehicle) idea. The performance of the system hinges on the path planning for the AGV. Its main purpose is to find an optimal non-collision path for each automatic guide car from the pre-stored parking space to the target parking space.

Hutagalung et al. (2021) [11] uses a magnetometer for sensing if the parking slot is empty . It produces a high accuracy result but comes with High latency also .Magnetic distortion is very little under a vehicle and may be disturbed by other vehicles. To improve the performance of the WVDs, an IoT-Driven vehicle detection method that combines the heterogeneous data feature of the UWB channel with that of the magnetic signal is proposed in **Lou et al.(2020) [12].**

Table 1. Gives a brief overview of all the Related works together

Author	**Link of publis hed work**	**year**	**Data**	**Features**	**Model/ Sensors used**	**Results**	**Drawback s**

Sairam et al.	[3]	2020	Images and videos coming through cameras, PKlot dataset	Vehicle detection, Checks empty spaces, Ignores moving objects	Sensor = Camera, Model = CNN(2)	93% of accuracy for detecting vehicles	Night time,wind or fog can disrupt camera input, can cause commotion sometimes
Padmasiri et al.	[4]	2020	Images and video coming through cameras, PKlot dataset for ML	Vehicle detection, Slot status updation	Sensor = camera , Model = RetinaNet R101 + R-CNN	Faster implementation due to dual models	Visibility concerns still pertain, very less (25.48%) accuracy of occupied space detection.
Agarwal et al.	[5]	2021	IR sensors output for each slot	Slot occupancy, Mobile and cloud integration Automatic gates	IR sensors, WIFI and RFID for processing ,Arduino	Parking space use +25%	High intensity light problems, Initial setup costs
Patro et al.	[6]	2020	InfraRed sensors at each slot	Slot occupancy and mobile app integration, Separate places for 2 and 4 wheelers	IR sensors , RFID	40% reduction in parking searching time	High intensity light problems, Initial setup costs
Kanan et al.	[7]	2020	Location of slots already at the cloud that is matched with that of vehicles	Automatic billing , parking spot realtime updation	VTD(2), RFID, GNSS(3), YOLOv3	Enhanced billing accuracy and automation	Too much dependent on internet, considerable chance of data mismatch of occupied slots
Thakre et al.	[8]	2021	Pre registered user database, data fetched in realtime by RFID readers	Seamless parking for registered user, App interface for parking space check	RFID tags , RFID users, Proximity sensors	Parking fees automatic deduction from user account possible	Single time users may not want to get registered .

Patil et al.	[9]	2023	Visual input from cameras	Identifies empty slot and show result directly on app	CNN algo for checking empty spaces for vehicle	Looks for small spaces for 2 wheelers also	Cannot effectively handle large places , no backup if camera is disrupted somehow
Wang et al.	[10]	2020	Already marked parking slots locations	Helps find a short and optimal path to an empty slot	Improved versions of ACO and GA(4) used	Drivers dont even have to look for their slots now	Sometimes an Infinite loop may form, Separate Entry exit not possible
Hutagalung et al.	[11]	2021	Magneto meter inputs	Checks for empty slots and updates data in both app and LCD	HMC5883L Magnetometer as sensor and wireless transceivers for communication	100% accuracy in open , closed , vehicle on off -Each case	The updating time for LCD reaches 40 seconds and 25 for app , High initial setup costs
Lou et al.	[12]	2020	Information of all parking spaces	Checks for empty slots and updates data on App	Fusion of RF based WVD(5) and Magnetometers	98% accuracy of detection , WVD is used if magnetometers behaves abnormally	High Energy consuptions and initial setup costs

(1) Convolutional Neural Networks (CNNs)

(2) Vehicle Transceiver device(VTD) : The VTD is designed to detect when a vehicle enters a parking area. It operates using a radio frequency (RF) field transmitted by a parking enable device (PED). Its attached to the car instead of parking lot. These vehicle detectors mainly include the inductive loop , acoustic sensor , infrared sensor or ultrasonic sensor .

(3) GNSS : global navigation satellite system

(4) ACO : Ant colony optimization

GA: Genetic algorithm

(5) WVD : Wireless Vehicle Detectors , they need robust mathematical algorithms for having desired accuracy.

Few common Drawbacks in the above mentioned projects are :

- **Environmental Sensitivity**:
  Many systems [3]-[6] that depends on cameras or infrared sensors have issues due to weather conditions such as fog, wind, High-intensity light(for IR specially), or low visibility during night time.

- **High Initial Setup Costs**:
  Several projects [5][6][11][12] mention high costs for setting up cameras, IR sensors, RFID readers, and other advanced Technologies. We have YOLO algorithm just at the entry gates for vehicle detection and number plate reading.

- **Technology Dependency**:
  Systems[7][8] depending only on internet connectivity or cloud-based data may face issues like data mismatches, delays and abnormal behaviours at odd times.

- **Limited Scalability**:
  Some solutions, particularly those using cameras [3][4][7][9] struggle to handle large-scale parking spaces or provide effective solutions for bigger parking areas.

- **Slow Response Times**:
  Certain systems[11] experience delays in updating information to users, either on apps or LCD displays, which can cause Inconvenience to the users.

- **Energy Consumption**:
  Systems with high energy demands [12] ,especially those using multiple sensors or wireless Transceivers, may require continuous power which increases the operational costs.

- **User Registration**:
  Solutions that require user registration[8] for seamless parking may not be okay to single-time users or visitors.

Project Description

Overview of the System

Smart parking system – three major components: a camera that monitors the approach of vehicles, some ultrasonic sensors to track free parking spots, and a Wokwi simulation to map the total scenario. These pieces all contribute in some way to ensure that the system remains operational[20].

Camera with YOLO Detection:

A camera is installed at the entry point of the parking lot, specifically on the barricade. The camera implements the YOLO algorithm that informs if the detected approaching object is a car, or not YOLO has great potential for real-time object detection, which is why it is adopted in this study. If the camera identifies the object as a car, it sends a signal (1) to the Wokwi simulator through the ThingSpeak cloud[14]. Otherwise, it sends a signal (0) which emphasizes that the detected object is not a vehicle and do not need any further action.

Ultrasonic Sensors for Slot Monitoring:

In our simulation, we have four parking slots, each using an ultrasonic sensor[16] paired with an ESP32 microcontroller. These sensors detect the distance to any object in front of them. The distance should be greater than 100 cm to consider the slot free. Otherwise, it is marked as occupied. These sensors allow real-time monitoring of the parking slots, making it such that the system is aware of exactly which slots are vacant at any given time.

The IOT simulator

We are using wokwi IOT simulator which combines the YOLO and the ultrasonic sensors data, is where main actuators are running. When the YOLO camera sends the cloud a "car detected" signal (1), the system comes to check the first parking slot status from Slot 1 till Slot 4 in numerical order. The parking slot that becomes available first is now determined, and the system communicates this to the LCD screen [17] by displaying a message indicating the slot number. When the parking lot is completely occupied, the LCD screen displays a "Space Full" message. We simulate a barricade that acts as the physical barrier at the entry of the parking area by using a servo motor. if an empty parking space is found and the corresponding slot number is displayed on the screen, the servo motor will then rotate to lift the barricade allowing the car to pass through.

Table 2. Implementation Table for Automated Parking System

Component	**Model**	**Use**
Camera	YOLOv8	Detects vehicles approaching the parking area and captures license plates.
Node MCU	ESP8266	Controls gate operations based on slot availability, integrates with cloud data.
Ultrasonic Sensor	HC-SR04	Checks if parking slots are vacant or occupied, sending status updates to the MCU.
LCD Display	I2C LCD	Displays available slot numbers or 'Space Full' message at the parking entry.
Cloud Platform	ThingSpeak	Receives and stores vehicle entry/exit data from the YOLO model for tracking.
Simulation	Wokwi	Visualizes the system's complete setup, integrating all IoT components for testing.
OCR	Optical Character Recognition (OCR)	Captures license plate details for vehicle tracking and billing.

The above table provides a concise overview of each component in the automated parking system, detailing the models used and their specific functions in enhancing efficiency.

Algorithm/Pseudocode for Automated Parking System Using YOLOv8

Inputs:

- Ultrasonic sensors for each parking slot to measure distance
- Switch to detect incoming cars using YOLO (object detection)
- LCD display to show parking slot availability
- Servo motor for barricade control

Output:

Displays available parking slot on LCD and controls servo to open/close barricade

Initialize Components:

Define and *set up* the LCD display.

Define and attach servo motor to Servo_pin, *set* initial position to 90° (closed).

Define pins for switch, ultrasonic sensor trigger (SensorTRG[i]), and echo pins (SensorECH[i]) for each parking slot.

Set avlSlot to represent the total number of parking slots.

Slot Setup (slotsetup):

For each parking slot i from 1 to avlSlot:

Assign SensorTRG[i] = 2*i + 1.

Assign SensorECH[i] = 2*i.

Setup Phase:

Set switchPin as input.

Attach servo to Servo_pin and *set* initial position at 90° (closed).

Initialize serial communication.

Call slotsetup() to configure each slot’s trigger and echo pins.

Set LCD display to starting screen.

Read Distance (readDistance):

```
For a given slot i:
Set SensorTRG[i] to LOW for 2 µs.
Set SensorTRG[i] to HIGH for 10 µs, then back to LOW.
Measure duration on SensorECH[i] to calculate distance.
Return calculated distance.
```

YOLO Detection Request (requestYOLO):

```
Return digital reading of switchPin to check for car presence.
```

Find Available Parking Slot (parkingSlot):

```
For each slot i from 1 to avlSlot:
If readDistance(i) > 100 cm, return slot i as available.
If no slot is available, return -1.
```

Main Loop:

```
Enable LCD backlight, set cursor to first row.
If requestYOLO() is true (car detected):
    Check for Available Parking Slot:
    Call parkingSlot() to find an available slot.
    If a slot is available:
         Open barricade by setting servo to 0°.
         Display “Slot No. [slot number]” on LCD, wait 3 seconds.
         Close barricade by setting servo back to 90°, wait 4 seconds.
    If no slots are available:
         Display "SPACE FULL" on LCD for 4 seconds.
If no car is detected:
```

Display "CAR PARKING" on LCD for 4 seconds.

Wait 500 ms before repeating the loop.

YOLO Object Detection Process

In this project, we developed two models: a vehicle detection model using YOLOv8 and a license plate detection model utilizing PyTorch. For the vehicle detection model, we began by training it on the KITTI vehicle dataset. After installing the necessary libraries, including ultralytics for YOLOv8 and wandb for experiment tracking, we logged into wandb using our API key. We organized the dataset in the required YOLO format, structuring it into separate folders for training and validation images along with their corresponding label files. A data.yaml configuration file was created to specify the paths and class names. We initialized the YOLOv8 model, selecting the YOLOv8n variant for efficiency. With wandb configured to monitor the training process, we set various training parameters, including the number of epochs, batch size, image size, and device. Executing the training script allowed the model to train while automatically logging performance metrics to wandb. Upon completion, we finished the wandb session to conclude tracking.

Simultaneously, we developed a license plate detection model and trained it on Kaggle. We first configured PyTorch in our local environment to ensure all dependencies were correctly installed. After organizing the dataset of vehicle images with annotated license plates into the appropriate structure, we utilized Kaggle's computing resources for efficient training. Once the model training was complete, we implemented an inference phase, loading the trained model to process new images for license plate detection. This involved analyzing the results to evaluate the model’s accuracy and performance in real-world scenarios. Through this comprehensive workflow, which integrated both vehicle and license plate detection systems, we effectively created functional and efficient solutions for vehicle identification and license plate recognition.

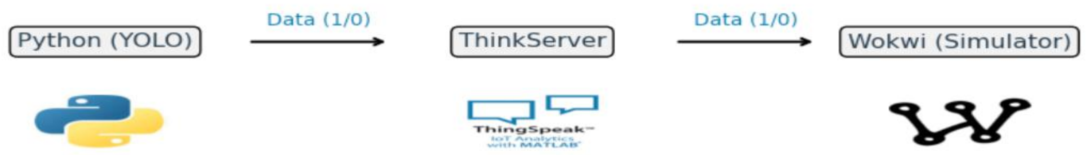


Fig 1. Data flow from YOLO to Cloud and finally to Wokwi

Components:

1. <u>Custom YOLOv8 Model</u>

- Annotated the text files in the yolo required format(<class_id><x_center><y_center><width> <height>)

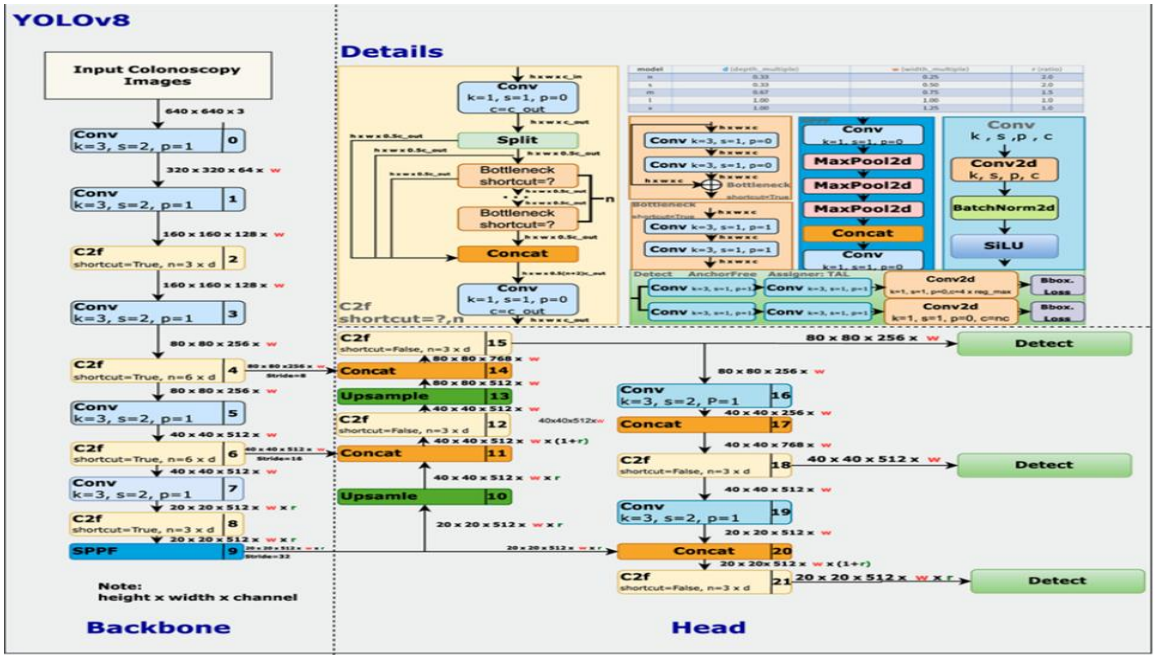


Fig 2. YOLOv8 Model: Simplified Overview of How It Detects Vehicles in Images

- Train a YOLOv8 model on the KITTI dataset to detect vehicles and pedestrians with accuracy of 94% and mAP of 94.61%.

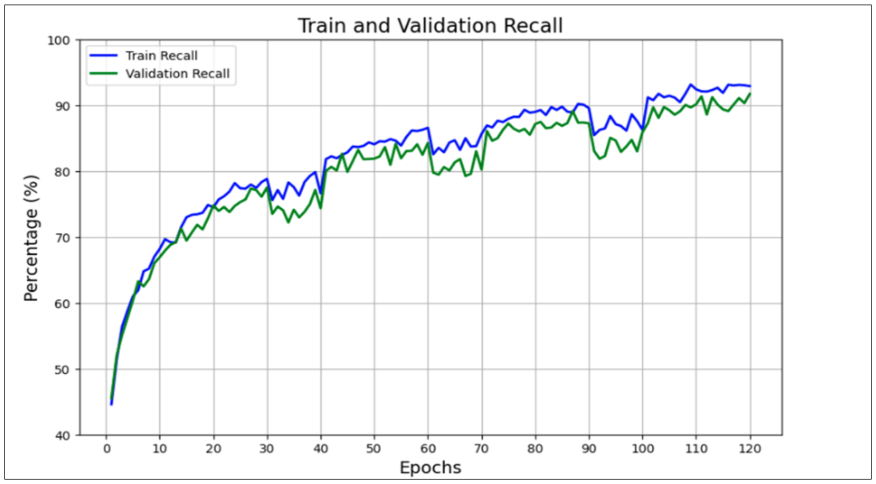


Fig 3. Training and validation recall curve

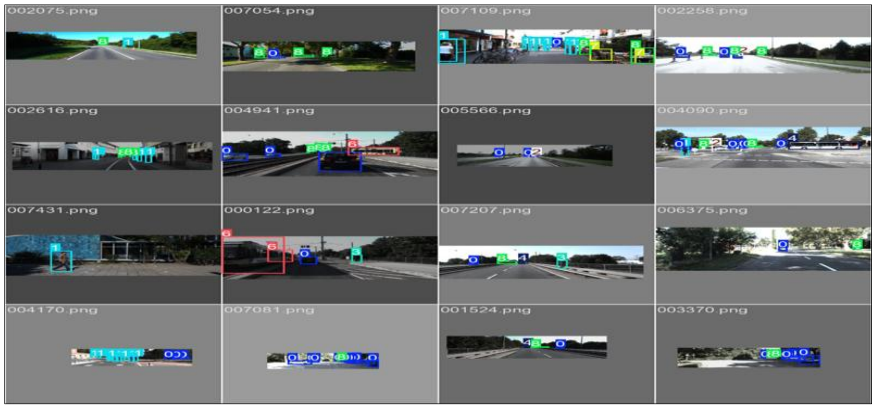


Fig 4. Realtime detection for various Scenarios

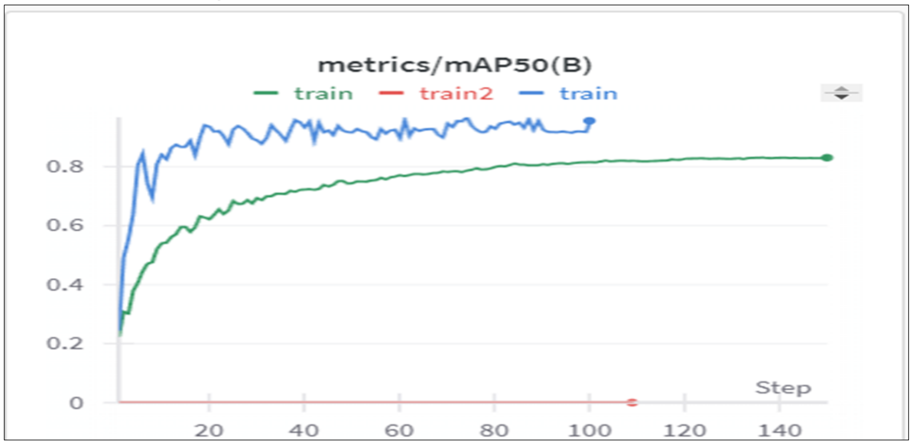


Fig 5. Training vs. validation mAP@50

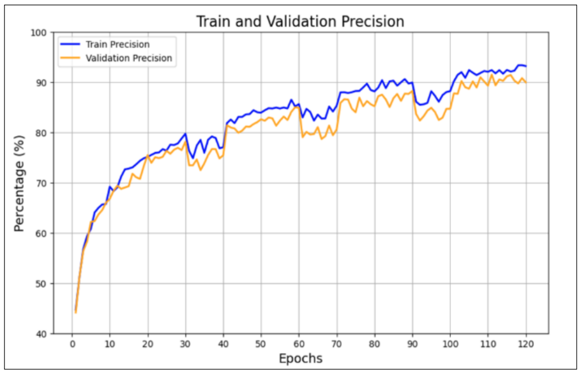


Fig 6. Precision over epochs

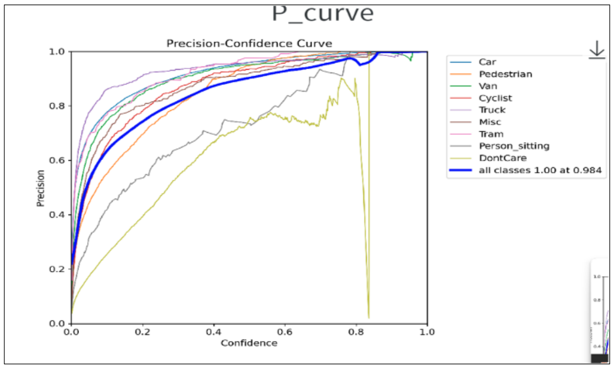
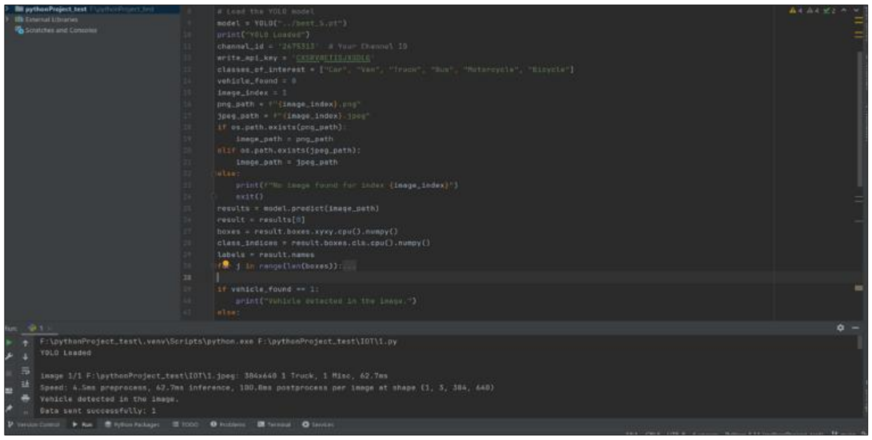


Fig 7. Precision vs confidence curve for the YOLO model over different classes

2. <u>Real-time Object Detection</u>:

 - Implement a real-time video feed (using a webcam or video file) where the YOLOv8 model will process frames to detect objects

- If Vehicle is found in front of the barricades then it will pass the data 1 to the server .



```
# Load the YOLO model
model = YOLO("../best_5.pt")
print("YOLO Loaded")
channel_id = '2475313'  # Your Channel ID
write_api_key = 'CKSRV8ETI5JX8DIG'
classes_of_interest = ["Car", "Van", "Truck", "Bus", "Motorcycle", "Bicycle"]
vehicle_found = 0
image_index = 1
png_path = f"{image_index}.png"
jpeg_path = f"{image_index}.jpeg"
if os.path.exists(png_path):
    image_path = png_path
elif os.path.exists(jpeg_path):
    image_path = jpeg_path
else:
    print(f"No image found for index {image_index}")
    exit()
results = model.predict(image_path)
result = results[0]
boxes = result.boxes.xyxy.cpu().numpy()
class_indices = result.boxes.cls.cpu().numpy()
labels = result.names
for j in range(len(boxes)):...

if vehicle_found == 1:
    print("Vehicle detected in the image.")
else:
```

```
F:\pythonProject_test\.venv\Scripts\python.exe F:\pythonProject_test\IOT\1.py
YOLO Loaded

image 1/1 F:\pythonProject_test\IOT\1.jpeg: 384x640 1 Truck, 1 Misc, 62.7ms
Speed: 4.5ms preprocess, 62.7ms inference, 100.8ms postprocess per image at shape (1, 3, 384, 640)
Vehicle detected in the image.
Data sent successfully: 1
```

Fig 8. Internal Python code Snippet used for detection of vehicle and sending signal to cloud accordingly

Table 3.Performance Metrics For YOLO Vehicle and Pedestrian Detection

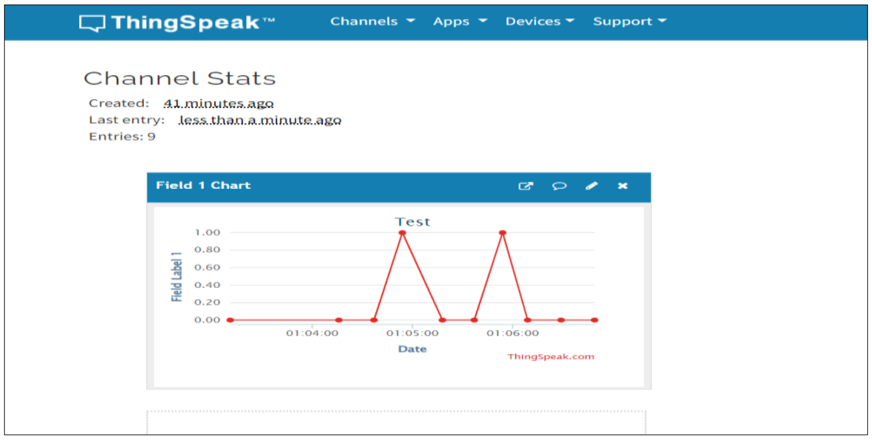
Metric	Accuracy
Accuracy	94%
Precision	95%
Recall	95.7%
F1-Score	95.34%
IoU	0.83

mAP@0.50	94.61%

3. Data Transfer to ThingSpeak:

- Set up a ThingSpeak channel to store detection data.

- Use the ThingSpeak API to send data from our detection system to the cloud server when a vehicle is detected[18].



Channel Stats

Created: 41 minutes ago

Last entry: less than a minute ago

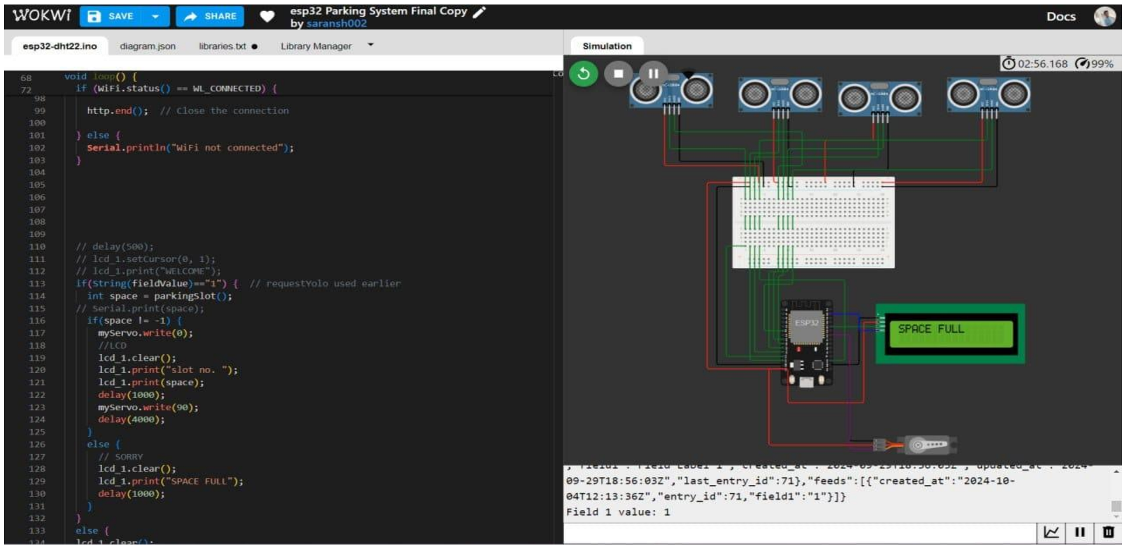
Entries: 9

Fig 9. Real-time data chart from ThingSpeak, showing field values over time with recent data entries

4. NodeMCU and Wokwi Integration:

- Create a Wokwi simulation [21] to represent the NodeMCU.
- Use the NodeMCU to retrieve data from the ThingSpeak cloud server.
- The NodeMCU is activated only when the field value it receives from the ThingSpeak cloud is 1.
- ThingSpeak server is enabled to send 1 only when it receives 1 from our python program.
- Overall, the NodeMCU (i.e. the vacant slot checking) is started only when there are any vehicles arriving at the gate of parking. Therefore, saving considerable amount of energy.

Have a look at the following figures 10 and 11 for better understanding.



```
void loop() {
  if (WiFi.status() == WL_CONNECTED) {

    http.end();  // Close the connection

  } else {
    Serial.println("WiFi not connected");
  }






  // delay(500);
  // lcd_1.setCursor(0, 1);
  // lcd_1.print("WELCOME");
  if(String(fieldValue)=="1") {  // requestYolo used earlier
    int space = parkingSlot();
  // Serial.print(space);
    if(space != -1) {
      myServo.write(0);
      //LCD
      lcd_1.clear();
      lcd_1.print("slot no. ");
      lcd_1.print(space);
      delay(1000);
      myServo.write(90);
      delay(4000);
    }
    else {
      // SORRY
      lcd_1.clear();
      lcd_1.print("SPACE FULL");
      delay(1000);
    }
  }
  else {
  lcd_1.clear();
```

```
09-29T18:56:03Z","last_entry_id":71},"feeds":[{"created_at":"2024-10-
04T12:13:36Z","entry_id":71,"field1":"1"}]}
Field 1 value: 1
```

Fig. 10 The NodeMCU starts to work only when it receives the field value 1

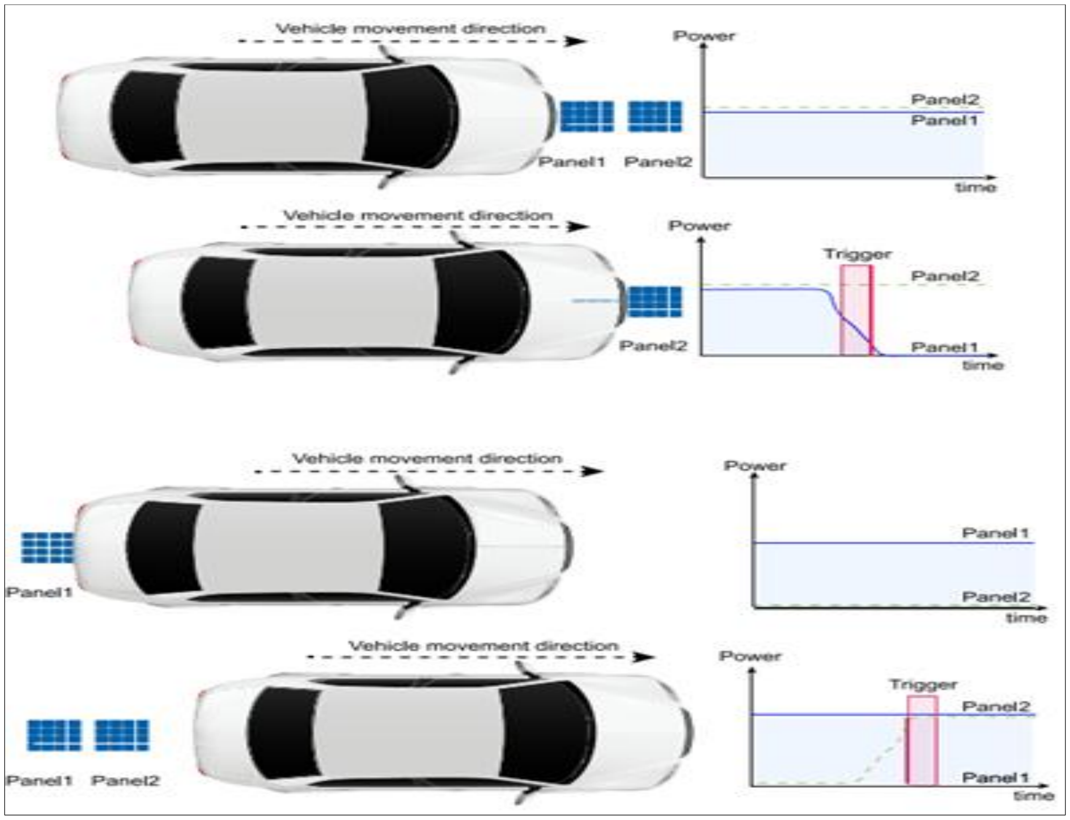


Fig 11. Illustrations of a vehicle traversing sensor panels, with graphs depicting power output and trigger events based on the vehicle's direction

Working example

This flowchart shows how a smart parking system works using cameras, sensors, and a small computer (Node MCU). It starts by using a camera to check if a vehicle or a person is present. If it's a vehicle, sensors check if there is an empty parking space. If there is space, the system opens the gate. If not, it shows "Parking Full." The system also reads the vehicle's license plate using OCR (Optical Character Recognition) and saves the information in a database for record-keeping.

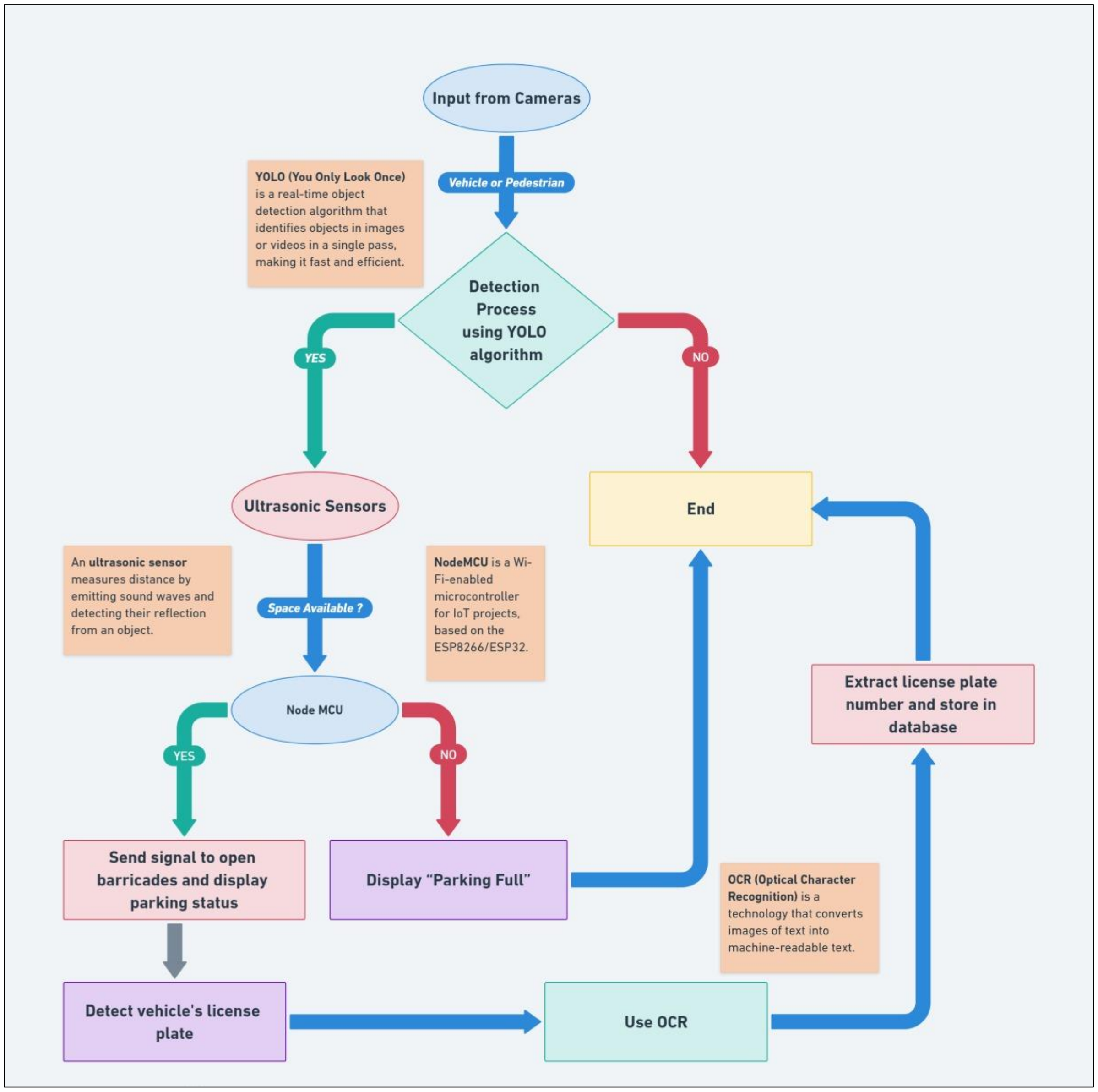
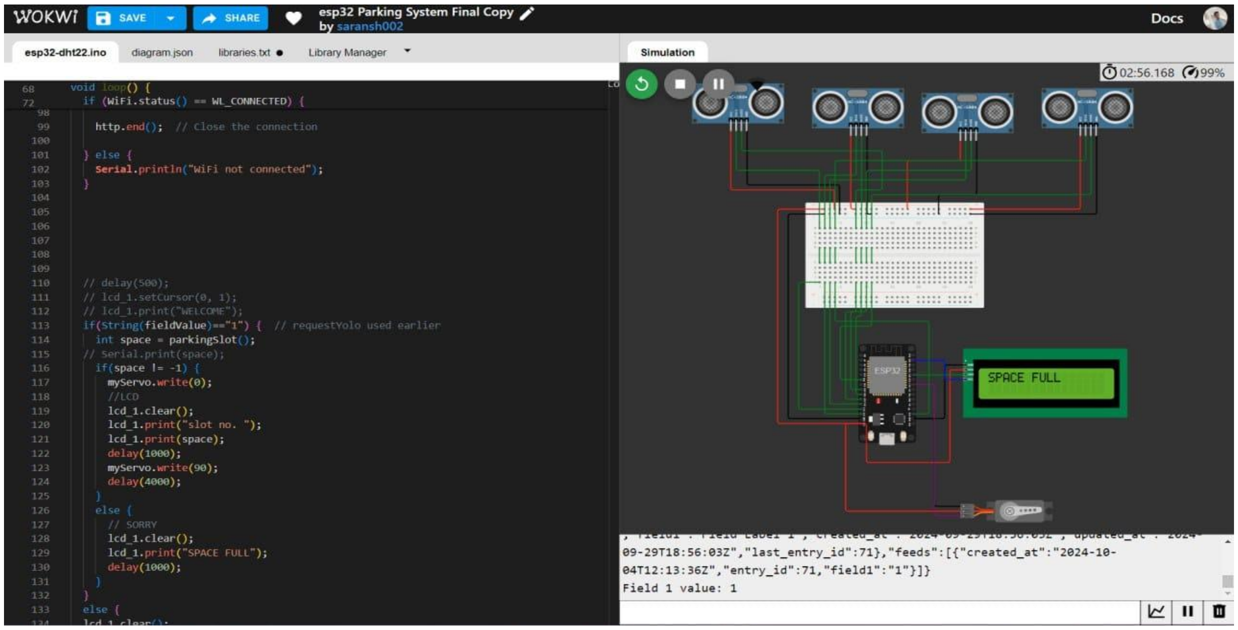

Input from Cameras
Vehicle or Pedestrian
YOLO (You Only Look Once) is a real-time object detection algorithm that identifies objects in images or videos in a single pass, making it fast and efficient.
Detection Process using YOLO algorithm
YES
NO
Ultrasonic Sensors
End
An ultrasonic sensor measures distance by emitting sound waves and detecting their reflection from an object.
Space Available ?
NodeMCU is a Wi-Fi-enabled microcontroller for IoT projects, based on the ESP8266/ESP32.
Extract license plate number and store in database
Node MCU
YES
NO
Send signal to open barricades and display parking status
Display "Parking Full"
OCR (Optical Character Recognition) is a technology that converts images of text into machine-readable text.
Detect vehicle's license plate
Use OCR

Fig 12. flowchart of smart parking system using YOLOv8 , ultrasonic sensors and Node MCU

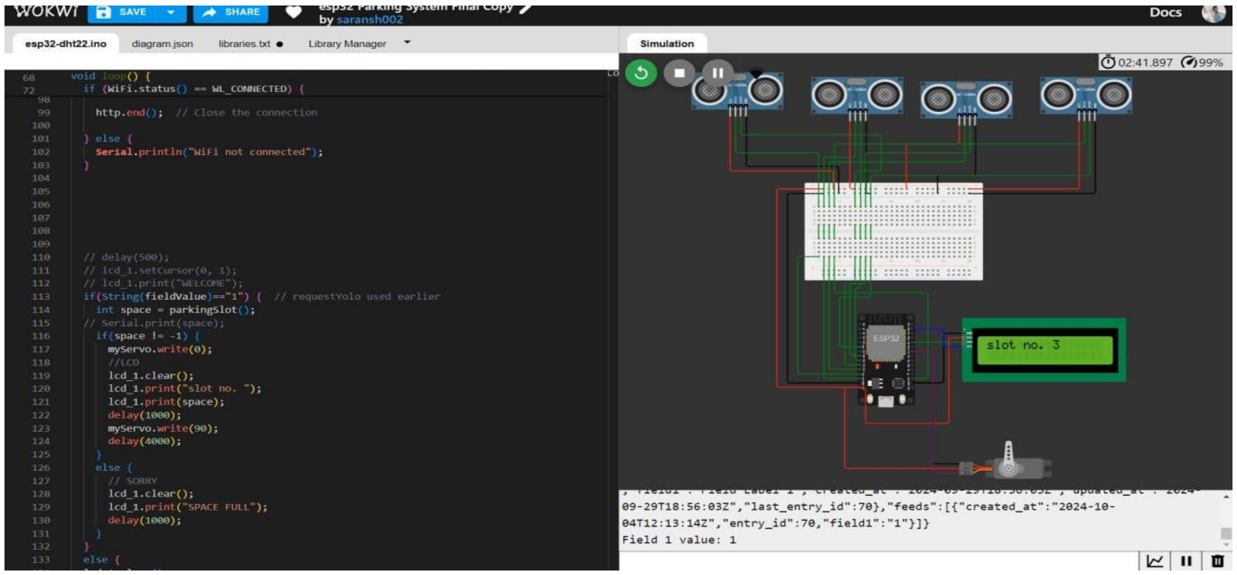
As we can see From Fig. 12 that the Node MCU waits for the confirmation from the ThingSpeak cloud server to trigger the further process of empty slot detection . The following Figures 12(a) and (b) shows the response of the Actuators , i.e. Servo motor and the LCD screen based on status of slots:



```
void loop() {
  if (WiFi.status() == WL_CONNECTED) {

    http.end();  // Close the connection

  } else {
    Serial.println("WiFi not connected");
  }






  // delay(500);
  // lcd_1.setCursor(0, 1);
  // lcd_1.print("WELCOME");
  if(String(fieldValue)=="1") {  // requestYolo used earlier
    int space = parkingSlot();
  // Serial.print(space);
    if(space != -1) {
      myServo.write(0);
      //LCD
      lcd_1.clear();
      lcd_1.print("slot no. ");
      lcd_1.print(space);
      delay(1000);
      myServo.write(90);
      delay(4000);
    }
    else {
      // SORRY
      lcd_1.clear();
      lcd_1.print("SPACE FULL");
      delay(1000);
    }
  }
  else {
```

```
09-29T18:56:03Z","last_entry_id":71},"feeds":[{"created_at":"2024-10-
04T12:13:36Z","entry_id":71,"field1":"1"}]}
Field 1 value: 1
```

Fig 13(a). Servo motor does not work when all the Slots Are Occupied , i.e. Gates does not open



```
void loop() {
  if (WiFi.status() == WL_CONNECTED) {

    http.end();  // Close the connection

  } else {
    Serial.println("WiFi not connected");
  }






  // delay(500);
  // lcd_1.setCursor(0, 1);
  // lcd_1.print("WELCOME");
  if(String(fieldValue)=="1") {  // requestYolo used earlier
    int space = parkingSlot();
  // Serial.print(space);
    if(space != -1) {
      myServo.write(0);
      //LCD
      lcd_1.clear();
      lcd_1.print("slot no. ");
      lcd_1.print(space);
      delay(1000);
      myServo.write(90);
      delay(4000);
    }
    else {
      // SORRY
      lcd_1.clear();
      lcd_1.print("SPACE FULL");
      delay(1000);
    }
  }
  else {
```

```
09-29T18:56:03Z","last_entry_id":70},"feeds":[{"created_at":"2024-10-
04T12:13:14Z","entry_id":70,"field1":"1"}]}
Field 1 value: 1
```

Fig 13(b). Servo motor works only when at least one of the spaces is unoccupied , i.e. Gates Open

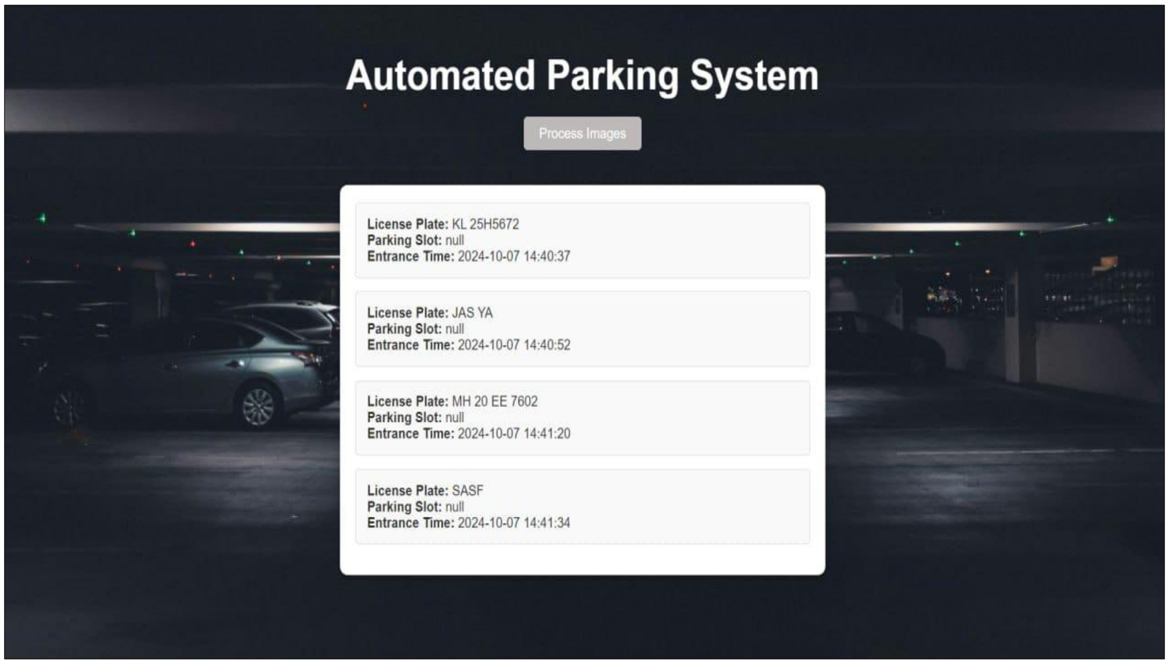
Users Data Management (Integrated Website)

We have also implemented a website to check the slot status with details of the user occupying the slot. The details include the vehicle number, slot number, and entry time of the vehicle. This information helps ensure the security of users, vehicles, and the parking area as well.

This data can also be used for research purposes in the future to improve the smoothness of operations, estimate peak times, enable efficient billing, optimize space, and allow for pre-booking of slots.

The Process of Collecting Data is as Follows:

1. YOLO detects the vehicle and extracts the number plate of the vehicle using OCR. This data is then sent to the server (FLASK).
2. The data is forwarded to Thingspeak, which relays it to an ESP32 in our IoT simulator (WOKWI).
3. When the vehicle is parked in a slot, the slot number is returned to the FLASK server via Thingspeak, facilitated by the ESP32.
4. Finally, all the data is displayed on the webpage.



License Plate: KL 25H5672
Parking Slot: null
Entrance Time: 2024-10-07 14:40:37

License Plate: JAS YA
Parking Slot: null
Entrance Time: 2024-10-07 14:40:52

License Plate: MH 20 EE 7602
Parking Slot: null
Entrance Time: 2024-10-07 14:41:20

License Plate: SASF
Parking Slot: null
Entrance Time: 2024-10-07 14:41:34

Fig 14. The interface of webpage that stores the data .

Comparision with the other related works mentioned

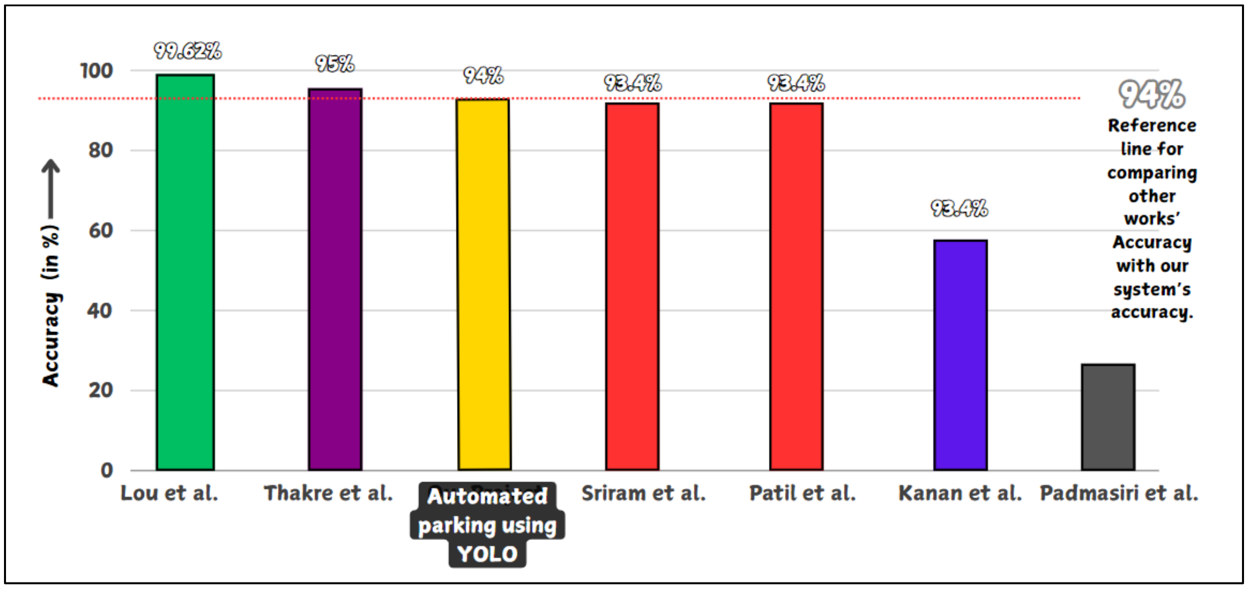


*Fig 15. A Bar Graph comparing Accuracy of different systems with ours**

The projects [5][6][10][11] that don't use any machine learning or AI models. Their detection accuracy have not been considered because these systems rely on simpler methods like proximity sensors or algorithms with fixed outputs, meaning they always produce the same results under certain conditions. These methods work well in controlled environments, but they aren't as flexible in real-world scenarios. Moreover the results have not be mentioned properly in the reports , hence they are not in the graph.

The graph shows that YOLOv8, which we used for object detection, achieves a 94% accuracy. This is significant because it's comparable to many of the other systems and even better than a few. Unlike systems that rely on predefined outputs, YOLOv8 uses AI to detect objects in real-time, adapting to different situations and conditions. This shows that our approach is very reliable, even when compared to other models that also use AI or a combination of sensors.

Overall, this comparison highlights the strength of our system in terms of accuracy, proving that YOLOv8 is well-suited for complex tasks like automated parking.

Conclusion

"Automated Parking System using YOLO" is an pretty effective design to use the potential of IoT and deep learning technologies to develop functionality and reliability in traditional parking systems.

YOLO does the vehicle detection; the optical character recognition determines number plates with minimal human intervention, thus effective tracking and management of the vehicles. This means that with the integration of Node MCU together with the ultrasonic sensors and the cloud-based database, the parking slots run in **real time**. Therefore, high accuracy metrics generated by the YOLO algorithm for the detection of vehicles are highly efficient that help us lower the amount of queues outside and inside parking areas.

Actually, it is a friendly user's interface in presenting real-time information regarding parking availability to the drivers, which further enhances experience for the users. Waiting time has been reduced and overall working operational efficiency improved because the system ensures automatic entry and exit.

Future Work

Advanced analytics for parking patterns analysis and forecasting, along with estimation of the time of peak demand for spaces, could be given as further directions towards development and enhancement of Automated Parking System. This would provide scope for dynamic pricing and optimum space allocation. Reservation of parking through a mobile application capable of making online reservations can also be included to make the reservation of parking before the user comes at station.

With the above features, for example, license plate whitelisting for regular users or a payment gateway, also for automated billing in the system, this would be possible and make it much smoother. Also, introducing electric vehicle charging stations in the parking layout would add value to the system and make it even more environmentally friendly to meet the increased demand for infrastructure for electric vehicles. Lastly, continuous improvement in the YOLO model through continuos training for many various datasets will further improve the detection accuracy and robustness and make the system robust to different parking environments. Conclusion Finally, notable achievements both in IoT and AI offer vast prospects for further development. Thus, the Automatic Parking System could continue being relevant and efficient even in an always changing urban system.

References

[1] https://ijcrt.org/papers/IJCRT2312302.pdf

[2] https://link.springer.com/chapter/10.1007/978-3-642-25453-6_61

[3] B. Sairam, A. Agrawal, G. Krishna and S. P. Sahu, "Automated Vehicle Parking Slot Detection System Using Deep Learning," 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2020, pp. 750-755, doi: 10.1109/ICCMC48092.2020.ICCMC-000140.

[4] Padmasiri, Heshan, Ranika Madurawe, Chamath Abeysinghe, and Dulani Meedeniya. "Automated vehicle parking occupancy detection in real-time." In *2020 Moratuwa engineering research conference (MERCon)*, pp. 1-6. IEEE, 2020.

[5] Y. Agarwal, P. Ratnani, U. Shah and P. Jain, "IoT based Smart Parking System," *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, 2021, pp. 464-470, doi: 10.1109/ICICCS51141.2021.9432196.

[6] Patro, Sibo Prasad, Padmaja Patel, Murali Krishna Senapaty, Neelamadhab Padhy, and Rahul Deo Sah. "IoT based smart parking system: a proposed algorithm and model." In *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, pp. 1-6. IEEE, 2020.

[7] R. Kanan and H. Arbess, "An IoT-Based Intelligent System for Real-Time Parking Monitoring and Automatic Billing," *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, Doha, Qatar, 2020, pp. 622-626, doi: 10.1109/ICIoT48696.2020.9089589.

[8] M. P. Thakre, P. S. Borse, N. P. Matale and P. Sharma, "IOT Based Smart Vehicle Parking System Using RFID," *2021 International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, India, 2021, pp. 1-5, doi: 10.1109/ICCCI50826.2021.9402699.

[9] Patil, Mahesh, Tejas Rawale, Nivrutti Charode, Tanmay Raut, and Swati Chandurkar. "The Smart System for Detection of Parking Slot for Vehicles." *Available at SSRN 4655177* (2023).

[10] Wang, Xianwei, Hao Shi, and Chao Zhang. "Path planning for intelligent parking system based on improved ant colony optimization." *IEEE Access* 8 (2020): 65267-65273.

[11] Hutagalung, A. J. M., I. K. Sjamsudin, and D. P. Hutabarat. "IoT-Based Parking Monitoring System using Magnetometer as the Sensor." In *IOP Conference Series: Earth and Environmental Science*, vol. 794, no. 1, p. 012134. IOP Publishing, 2021.

[12] Lou, Liangliang, Qiang Li, Zengchao Zhang, Rong Yang, and Wei He. "An IoT-driven vehicle detection method based on multisource data fusion technology for smart parking management system." *IEEE Internet of Things Journal* 7, no. 11 (2020): 11020-11029.

[13] https://forum.arduino.cc/c/community/9 . Arduino Community Forum. “Community of general people who loves IOT, Here most of the problem faced in coding, connections in circuits are multually solved”

[14] https://www.mathworks.com/help/thingspeak/ thingSpeak documentation. “Elaborated ways to use cloud servers in IOT”

[15] https://nodemcu.readthedocs.io/en/release/ NodeMcu Documentaion. “Explained the working, Scope, Adavantages and disadvantages of using Node MCU in IOT”

[16] https://web.eece.maine.edu/~zhu/book/lab/HC-SR04%20User%20Manual.pdf User Manual for Ultrasonic Sensor

[17] https://projecthub.arduino.cc/arduino_uno_guy/i2c-liquid-crystal-displays-5eb615 I2C Liquid Crystal Dispaly Manual

[18] https://randomnerdtutorials.com/esp8266-nodemcu-thingspeak-publish-arduino/ “ESP8266 NodeMCU Publish Sensor Readings to ThingSpeak”

[19] https://www.researchgate.net/publication/350890091_Novel_IOT_Nodemcu_projects Harpreet Kaur Channi “Edition: 1 Publisher: LAP Germany Editor: LAP ISBN: 978-620-0-45806-3 “

[20] https://webbylab.com/blog/iot-based-smart-parking-system-step-by-step-guide-on-how-to-create-your-own-iot-solution/#:~:text=IoT%20Smart%20Parking%20System%3A%20Working%20Principles%20%26%20Architecture,-A%20smart%20parking&text=A%20smart%20parking%20system%20uses,like%20parking%20administrators%20and%20drivers. Smart parking System “Detailed Guide to IoT-Based Smart Parking System Development”

[21] https://docs.esp-rs.org/book/tooling/simulating/wokwi.html Wokwi Starting Guide “Use of simulator”

[22] https://www.researchgate.net/publication/377216968_A_Review_on_YOLOv8_and_Its_Advancements