SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY(Autonomous)

Batch NO:18

Project Name: License Plate Recognition System

Group Members: DUDDU JESHWIN(21781A0456)

DADAM SRIKANTH REDDY(21781A0450)

CHANDAN(21781A0451)

C.M.RAMESHREDDY(21781A0444)

LICENSE PLATE RECOGNITION SYSTEM

PROBLEM STATEMENT :

License Plate Recognition (LPR) systems are crucial in various applications such as traffic management, toll collection, parking management, and law enforcement. However, the implementation of such systems often poses challenges due to the complexity of image processing algorithms, hardware requirements, and real-time processing constraints.

The problem at hand is to design and develop a robust License Plate Recognition System utilizing Raspberry Pi, an affordable and widely accessible single-board computer. This system aims to accurately detect and recognize license plates from images or video streams captured by a camera connected to the Raspberry Pi.

SCOPE OF SOLUTION :

The scope of solutions for a license plate recognition (LPR) system using a Raspberry Pi can vary based on factors such as:

1. **Hardware Requirements**: Raspberry Pi is a low-cost, single-board computer, so the solution needs to be optimized to work within its hardware limitations, including processing power, memory, and connectivity options.
2. **Software Development**: The software stack will include components for image processing, optical character recognition (OCR), and possibly machine learning for improving recognition accuracy. This could involve libraries like OpenCV for image processing and Tesseract for OCR.
3. **Integration with Cameras**: Selecting compatible cameras with suitable resolution and frame rates is crucial. USB or Raspberry Pi camera modules can be used depending on the application requirements.
4. **Algorithm Complexity**: The system's algorithmic complexity should be tailored to run efficiently on the Raspberry Pi's hardware. This might involve optimizations such as reducing the resolution of incoming images, employing pre-processing techniques to enhance image quality, or using simpler but efficient recognition algorithms.
5. **Real-Time Processing**: Depending on the application, real-time processing capabilities may be required for immediate recognition and response. This necessitates efficient algorithms and hardware optimizations.

6. **Accuracy and Robustness**: The system should be capable of accurately recognizing license plates under various conditions such as different lighting conditions, angles, distances, and plate types. Robustness to factors like motion blur, occlusion, and plate degradation is also important.
7. **Data Storage and Retrieval**: The recognized license plate data needs to be stored and managed efficiently. This might involve databases or cloud services for storage and retrieval.
8. **User Interface**: Depending on the application, a user interface may be required for configuration, monitoring, and interaction with the system.
9. **Security and Privacy**: If handling sensitive data, security measures should be implemented to protect data integrity and privacy. This might involve encryption, access controls, and secure communication protocols.
10. **Scalability**: Consideration should be given to how easily the system can be scaled to handle increased loads or accommodate additional features in the future.
11. **Legal and Regulatory Compliance**: Ensure compliance with relevant laws and regulations regarding the use of LPR technology, especially concerning privacy and data protection.
12. **Deployment and Maintenance**: Procedures for deploying the system in the target environment and maintaining it over time should be established.

Overall, the scope of a license plate recognition system using a Raspberry Pi encompasses various aspects of hardware, software, algorithm development, integration, performance optimization, and compliance considerations tailored to the specific requirements of the application.

REQUIRED COMPONENTS TO DEVELOP SOLUTIONS :

HARDWARE COMPONENTS :

1.Raspberry

2.Camera

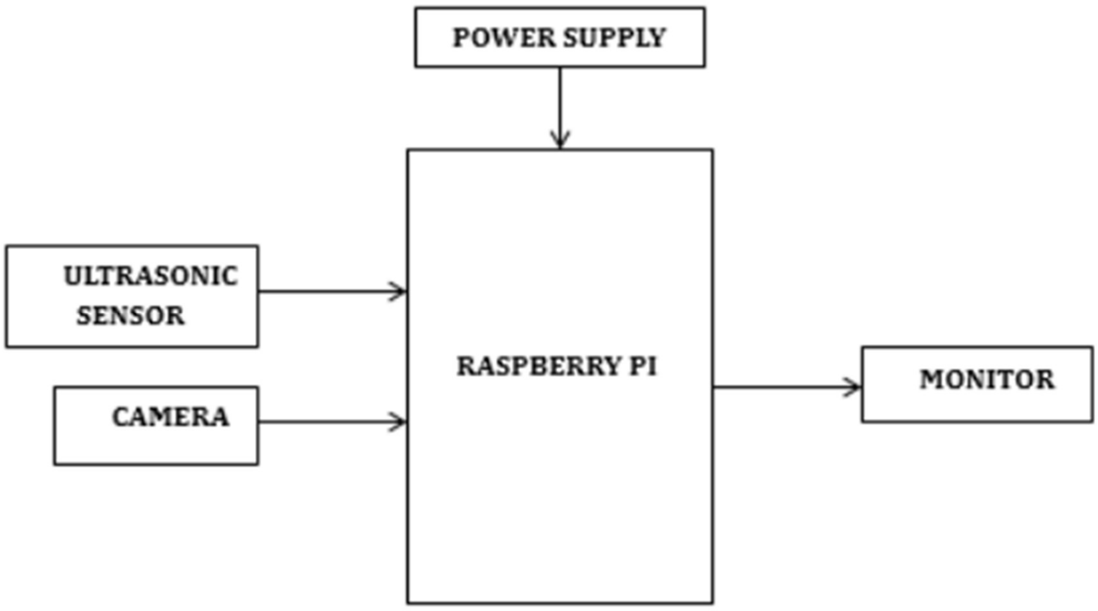
3.Power supply

4.storage

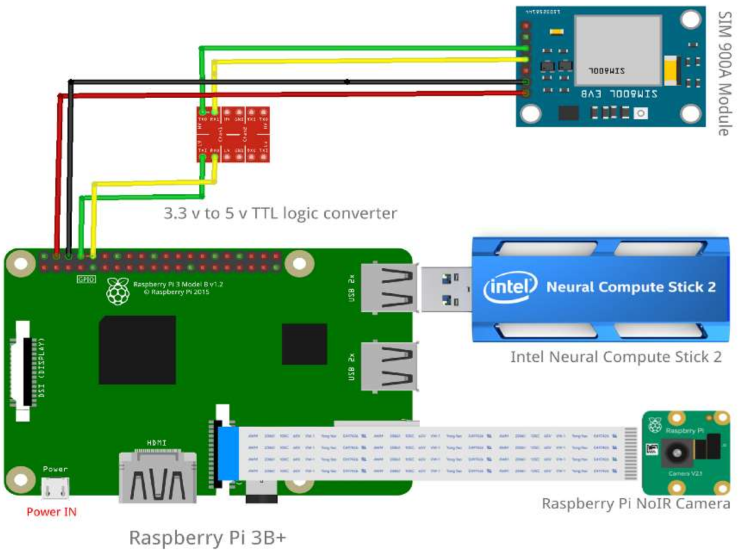
5.Enclosure

6.Mounting hardware

BLOCK DIAGRAM :



SIMULATED CIRCUIT :



CODE FOR SOLUTION :

```
import cv2

import pytesseract


# Path to Tesseract executable (Change this path according to your
installation)

 pytesseract.pytesseract.tesseract_cmd = '/usr/bin/tesseract'


   def detect_license_plate(image_path):

    # Load image

    image = cv2.imread(image_path)


    # Convert image to grayscale

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)


    # Apply image thresholding

    _, threshold = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY)


    # Find contours

    contours, _ = cv2.findContours(threshold, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)


    # Filter contours based on area

    contours = [cnt for cnt in contours if cv2.contourArea(cnt) > 1000]
```

```
# Sort contours by area
  contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]

  for contour in contours:
  # Get approximate polygon for contour
  epsilon = 0.05 * cv2.arcLength(contour, True)
  approx = cv2.approxPolyDP(contour, epsilon, True)

  # If contour has four corners (likely a license plate)
  if len(approx) == 4:
     x, y, w, h = cv2.boundingRect(approx)

     # Extract license plate region
     license_plate = gray[y:y+h, x:x+w]

     # Perform OCR on license plate region
     text = pytesseract.image_to_string(license_plate, config='--psm 6')

     # Print detected license plate text
     print("Detected License Plate:", text.strip())

     # Draw bounding box around license plate
     cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
     cv2.putText(image, text.strip(), (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
     break
```

```
    # Display result
    cv2.imshow('License Plate Detection', image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# Test the function with an example image
image_path = 'example_image.jpg'  # Replace with your image path
detect_license_plate(image_path)
```

CONCLUSION:

In conclusion, building a license plate recognition system using a Raspberry Pi requires careful consideration of hardware, software, algorithmic, and regulatory aspects to create a functional and efficient solution. While this technology offers great potential for various applications such as parking management, traffic monitoring, and security systems, it also presents challenges that need to be addressed through thorough design, implementation, and testing processes.