AMRITA VISHWA VIDYAPEETHAM

Title: Autonomous car with collision detection and avoidance

Team members:

Dheeraj-(CB.AI.U4AIM24109)

Sai Charan- (CB.AI.U4AIM24124)

Prem Siva Sai- (CB.AI.U4AIM24125)

Chiru Deep-(CB.AI.U4AIM24137)

Teachers:

Dr. Amrutha V

Dr. Snigdhatanu Archaya

Subject:

24AIM113: Introduction To NN, CNN, GNN,

24AIM114: Analog System Design.

Steps to connect esp32cam:

1. Install the “esp32(Espressif)” Package in Board Manager in the Arduino IDE.
2. Open file and go to preferences
   and add this links in the Additional boards manager URLs
   http://arduino esp8266.com/stable/package_esp8266com_index.json;

   https://dl.espressif.com/dl/package_esp32_index json.
3. Select AI Thinker ESP32_CAM Board and port.
4. Open files and select Examples->ESP32->Camera->CameraWebServer.
5. Enable CAMERA_MODEL_AI_THINKER in the code.

1. Package Of ESP32 In Arduino IDE by Expressif:

The "ESP32 by Espressif" package is an Arduino core package developed and maintained by Espressif Systems. It allows you to program ESP32 microcontrollers using the familiar Arduino IDE environment.

Purpose: To provide a complete software development platform for ESP32 boards.

- It supports built-in WiFi, Bluetooth/BLE, numerous peripheral interfaces, and a rich set of libraries and examples.

2. **#include "esp_camera.h"**

Purpose:
This is a specialized library provided by Espressif (This makes the ESP32) to interface with camera modules, especially for the **ESP32-CAM** board.
Use:

- Controls the camera hardware on the ESP32-CAM board.
- Initializes the camera and configures settings like resolution, pixel format, brightness, contrast, etc.
- Captures images (either single frames or streaming video).

3. **#include <WiFi.h>**

Purpose:
This is the standard **Wi-Fi library** for the ESP32 board. It enables the ESP32 to connect to Wi-Fi networks and act as a client or a server.

Uses:

- Connects the ESP32 to a Wi-Fi access point (WiFi.begin()).
- Provides network details like IP address

ESP 32 code

```
#include "esp_camera.h"
#include <WiFi.h>

//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//            Ensure ESP32 Wrover Module or other board with PSRAM is selected
//            Partial images will be transmitted if image exceeds buffer size
```

```
 //
 //            You must select partition scheme from the board menu that has at
least 3MB APP space.
 //            Face Recognition is DISABLED for ESP32 and ESP32-S2, because it
takes up from 15
 //            seconds to process single frame. Face Detection is ENABLED if
PSRAM is enabled as well

 // ===================
 // Select camera model
 // ===================
 //#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
 //#define CAMERA_MODEL_ESP_EYE  // Has PSRAM
 //#define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
 //#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
 //#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
 //#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
 //#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
 //#define CAMERA_MODEL_M5STACK_UNITCAM // No PSRAM
 //#define CAMERA_MODEL_M5STACK_CAMS3_UNIT  // Has PSRAM
 #define CAMERA_MODEL_AI_THINKER // Has PSRAM
 //#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
 //#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM
 // ** Espressif Internal Boards **
 //#define CAMERA_MODEL_ESP32_CAM_BOARD
 //#define CAMERA_MODEL_ESP32S2_CAM_BOARD
 //#define CAMERA_MODEL_ESP32S3_CAM_LCD
 //#define CAMERA_MODEL_DFRobot_FireBeetle2_ESP32S3 // Has PSRAM
 //#define CAMERA_MODEL_DFRobot_Romeo_ESP32S3 // Has PSRAM
 #include "camera_pins.h"

 // ===========================
 // Enter your WiFi credentials
 // ===========================
 const char *ssid = "premsiva";
 const char *password = "1234567890";

 void startCameraServer();
 void setupLedFlash(int pin);

 void setup() {
   Serial.begin(115200);
   Serial.setDebugOutput(true);
   Serial.println();

   camera_config_t config;
```

```
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.frame_size = FRAMESIZE_UXGA;
  config.pixel_format = PIXFORMAT_JPEG;  // for streaming
  //config.pixel_format = PIXFORMAT_RGB565; // for face detection/recognition
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;

  // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
  //                      for larger pre-allocated frame buffer.
  if (config.pixel_format == PIXFORMAT_JPEG) {
    if (psramFound()) {
      config.jpeg_quality = 10;
      config.fb_count = 2;
      config.grab_mode = CAMERA_GRAB_LATEST;
    } else {
      // Limit the frame size when PSRAM is not available
      config.frame_size = FRAMESIZE_SVGA;
      config.fb_location = CAMERA_FB_IN_DRAM;
    }
  } else {
    // Best option for face detection/recognition
    config.frame_size = FRAMESIZE_240X240;
#if CONFIG_IDF_TARGET_ESP32S3
    config.fb_count = 2;
#endif
  }
```

```
#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t *s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1);        // flip it back
    s->set_brightness(s, 1);   // up the brightness just a bit
    s->set_saturation(s, -2);  // lower the saturation
  }
  // drop down frame size for higher initial frame rate
  if (config.pixel_format == PIXFORMAT_JPEG) {
    s->set_framesize(s, FRAMESIZE_QVGA);
  }

#if defined(CAMERA_MODEL_M5STACK_WIDE) ||
defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

#if defined(CAMERA_MODEL_ESP32S3_EYE)
  s->set_vflip(s, 1);
#endif

// Setup LED FLash if LED pin is defined in camera_pins.h
#if defined(LED_GPIO_NUM)
  setupLedFlash(LED_GPIO_NUM);
#endif

  WiFi.begin(ssid, password);
  WiFi.setSleep(true);

  Serial.print("WiFi connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print("");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {
  // Do nothing. Everything is done in another task by the web server
  delay(10000);
}
```

The modifications that we have done is :

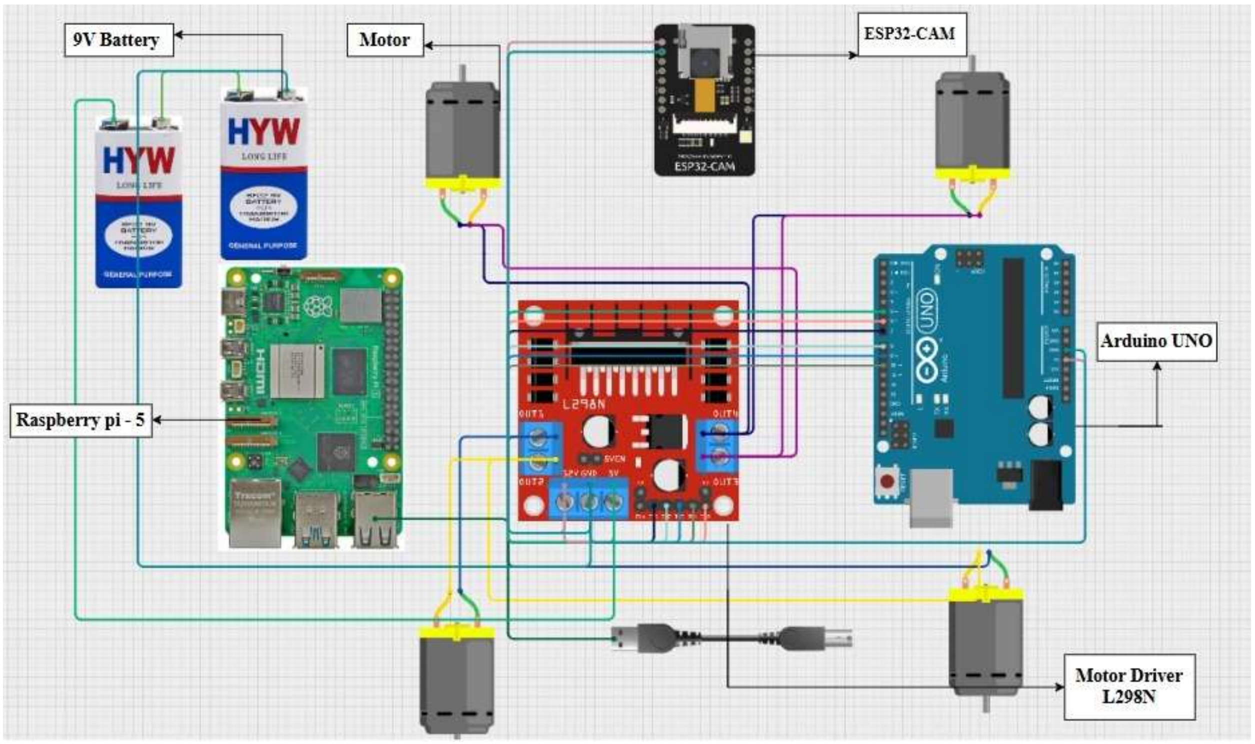
1. choosing better data transfer cable.
2. installing the drivers that are required.
3. we rectified error in the code

```
WiFi.begin(ssid, password);
  WiFi.setSleep(true);
%% in above code frist time there is false .we modified it to true.
```

4. ssid, password and WI-FI of system should connect to the same device.

5. installing the Esp 32 library in arduino IDE.

Model circuit diagram



Code of prototype with the interfaceing, motors,(used ardiuno uno and raspberry Pi), motor driver .

```
// Motor driver pins
#define ENA 5
#define ENB 6
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 10

// Speed constants
#define MAX_REGULAR_MOTOR_SPEED 80
#define MAX_MOTOR_ADJUST_SPEED 60

void setup() {
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  Serial.begin(9600);
  Serial.println("Ready for commands:");
  Serial.println("f = Forward, b = Backward, l = Left, r = Right, s = Stop");
}

void rotateMotor(int leftSpeed, int rightSpeed) {
  // Left motor direction
  if (leftSpeed > 0) {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
  } else if (leftSpeed < 0) {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
  } else {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
  }

  // Right motor direction
  if (rightSpeed > 0) {
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
  } else if (rightSpeed < 0) {
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
  } else {
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
  }

  // Apply speed (positive value only)
  analogWrite(ENA, abs(leftSpeed));
  analogWrite(ENB, abs(rightSpeed));
}

void loop() {
  if (Serial.available()) {
    char command = Serial.read();
    int leftSpeed = 0;
    int rightSpeed = 0;

    Serial.print("Received Command: ");
    Serial.println(command);

    switch (command) {
      case 'f': // Forward
```

```
        leftSpeed = MAX_REGULAR_MOTOR_SPEED;
        rightSpeed = MAX_REGULAR_MOTOR_SPEED;
        break;
      case 'b': // Backward
        leftSpeed = -MAX_REGULAR_MOTOR_SPEED;
        rightSpeed = -MAX_REGULAR_MOTOR_SPEED;
        break;
      case 'l': // Left turn
        leftSpeed = -MAX_MOTOR_ADJUST_SPEED;
        rightSpeed = MAX_MOTOR_ADJUST_SPEED;
        break;
      case 'r': // Right turn
        leftSpeed = MAX_MOTOR_ADJUST_SPEED;
        rightSpeed = -MAX_MOTOR_ADJUST_SPEED;
        break;
      case 's': // Stop
        leftSpeed = 0;
        rightSpeed = 0;
        break;
      default:
        Serial.println("Invalid Command");
        return; // Exit early without calling rotateMotor
    }

    rotateMotor(leftSpeed, rightSpeed);
  }
}
```

Steps involved connecting raspberry pi to VNC SERVER DESKTOP

1. Components Needed:

- Raspberry Pi
- Laptop
- Power cable
- LAN cable
- SD card
- SD card reader (USB)

2.Software Required:

- Raspberry Pi Imager

- Advanced IP Scanner
- PuTTY
- VNC Viewer

Step-by-Step Process

Step 1: Flash Raspberry Pi OS

- Use Raspberry Pi Imager to flash the OS onto your SD card.

Step 2: Enable SSH

- After flashing, open the boot partition of the SD card.
- Create a blank file named ssh (no file extension).

Step 3: Setup Hardware Connections

- Eject the SD card from your laptop.
- Insert it into the Raspberry Pi.
- Connect the LAN cable between the Raspberry Pi and your laptop.
- Connect the power supply to the Pi.

Step 4: Configure Network Sharing on Laptop

- Open Control Panel → Network and Internet → Network and Sharing Center.
- Go to the Ethernet connection → Properties → Internet Protocol Version 4 (IPv4).
- Check if any IP address is showing.

Step 5: Share Internet from Wi-Fi to Ethernet

- If no IP is shown, go to your Wi-Fi adapter → Properties → Sharing tab.
- Enable "Allow other network users to connect through this computer's internet connection."
- Set the shared connection to Ethernet.

Step 6: Confirm Ethernet IP

- Go back to Ethernet → Properties → IPv4.

- You should now see IP address: 192.168.137.1.

Step 7: Scan for Raspberry Pi IP

- Open Advanced IP Scanner.
- Set IP range: 192.168.137.0 to 192.168.137.254.
- Click Scan. Note the Pi's IP (e.g., 192.168.137.14).

Step 8: Connect via PuTTY

- Open PuTTY.
- Enter the Raspberry Pi's IP in the Host Name field.
- Click Open → Enter username (pi) and password (raspberry by default).

Step 9: Configure Raspberry Pi

- In terminal, type this code:

```
sudo raspi-config
```

- Go to Interfacing Options → Enable SSH and VNC.
- Go to System Options → Boot/Auto Login → Choose Desktop Autologin.
- Set a default browser.
- Again, go to Interfacing Options → Enable RPi Connect.
- Click Finish → Raspberry Pi will reboot.

Step 10: Reconnect via PuTTY

- After reboot, open PuTTY again and connect with the same IP.
- Log in again with username and password.

Step 11: Install and Start VNC

Use these commands to set up VNC:

Code	**Use**

which vncserver	Checks if VNC server is installed
sudo apt update	Updates the package list
sudo apt install realvnc-vnc-server realvnc-vnc-viewer	Installs VNC packages
sudo systemctl enable vncserver-x11-serviced	Enables VNC service on boot
sudo systemctl start vncserver-x11-serviced	Starts the VNC service
vncserver	Starts VNC server manually
sudo raspi-config	Opens configuration menu
sudo reboot	Reboots the Pi
sudo systemctl status vncserver-x11-serviced	Checks VNC service status

Step 12: Final VNC Setup

- If needed, run sudo raspi-config again and re-enable VNC.
- Reboot if required.

Step 13: Connect via VNC Viewer

- Open VNC Viewer on your laptop.
- Paste the Pi’s IP in the search bar.
- It will prompt for username and password.
- Enter credentials → Now you're accessing Raspberry Pi desktop remotely!

Raspberry Pi code for the object detection:

1. cv2 (OpenCV):

OpenCV is a library used for image and video processing tasks.With it, can read, write, and manipulate images and videos easily. It supports real-time object detection, face recognition, and more. It works with numpy arrays, making pixel-level operations easy.it is widely used in robotics, computer vision, and AI projects. Example: cv2.imread(), cv2.imshow(), cv2.VideoCapture()

2. numpy:

NumPy is a numerical computing library used for handling arrays and matrices. It allows fast mathematical operations on large datasets. Functions like mean, sum, reshape, and dot product are built-in.Used heavily in data science, machine learning, and scientific computing. NumPy arrays are more efficient than Python lists.
Example: np.array(), np.mean(), np.dot().

3.TensorFlow:

TensorFlow is an open-source library by Google for machine learning and deep learning.
It allows building and training neural networks for tasks like image classification and NLP.It supports CPUs, GPUs, and TPUs for scalable computing. TensorFlow also offers tools like Tensor Board for visualization. Commonly used with Keras (built-in API) for easy model design.
Example: tf.keras.Model, tf.constant(), tf.function()

4.serial (pySerial):

serial from the pySerial module is used for communication with serial devices like Arduino.
It allows reading and writing data through USB or COM ports. Useful in robotics, IoT, and embedded system projects. Can be used to send sensor data or control signals.
Requires setting port name and baud rate.
Example: serial. Serial('/dev/ttyUSB0", 9600), read(), write().

5.time

The time module provides functions related to time operations.
It can pause execution using sleep(), or measure time using time().
Useful for creating delays, benchmarking, or scheduling tasks.
Can also convert between time formats.
Common in automation, robotics, and logging events.
Example: time.sleep(1), time.time()

```
import cv2
```

```
 import numpy as np
 import tensorflow as tf
 import serial
 import time

 # Initialize Arduino serial connection (use correct port!)
 arduino = serial.Serial('/dev/ttyUSB0', 9600)  # Replace with your Arduino's
port
 time.sleep(2)

 # Load Faster R-CNN model
 detect_fn = tf.saved_model.load('faster_rcnn_model/saved_model')

 LABELS = {
     1: 'person', 2: 'bicycle', 3: 'car', 4: 'motorcycle', 5: 'bus',
     6: 'truck', 10: 'traffic light', 13: 'stop sign'
 }

 ESP32_STREAM = 'http://192.168.181.204:81/stream'
 cap = cv2.VideoCapture(ESP32_STREAM)

 if not cap.isOpened():
     print("Unable to connect to ESP32-CAM stream.")
     exit()
 print("Connected to ESP32-CAM stream...")

 GRID_SIZE = 10

 def create_grid(obstacles):
     grid = [[0 for _ in range(GRID_SIZE)] for _ in range(GRID_SIZE)]
     for (x, y) in obstacles:
         if 0 <= x < GRID_SIZE and 0 <= y < GRID_SIZE:
             grid[y][x] = 1
     return grid

 def a_star(start, goal, grid):
     from queue import PriorityQueue
     dirs = [(0,1), (1,0), (0,-1), (-1,0)]
     q = PriorityQueue()
     q.put((0, start))
     came_from = {start: None}
     cost_so_far = {start: 0}

     while not q.empty():
         _, current = q.get()
```

```
        if current == goal:
            break

        for dx, dy in dirs:
            nx, ny = current[0] + dx, current[1] + dy
            if 0 <= nx < GRID_SIZE and 0 <= ny < GRID_SIZE:
                if grid[ny][nx] == 1:
                    continue
                new_cost = cost_so_far[current] + 1
              if (nx, ny) not in cost_so_far or new_cost < cost_so_far[(nx,
ny)]:
                    cost_so_far[(nx, ny)] = new_cost
                    priority = new_cost + abs(goal[0]-nx) + abs(goal[1]-ny)
                    q.put((priority, (nx, ny)))
                    came_from[(nx, ny)] = current

    path = []
    node = goal
    while node and node != start:
        path.append(node)
        node = came_from.get(node)
        if node is None: return []  # No path found
    path.reverse()
    return path

def get_command_from_path(path):
    if len(path) < 1:
        return 'S'  # Stop
    next_node = path[0]
    dx = next_node[0] - 5
    dy = next_node[1] - 5
    if dx == 1: return 'R'
    elif dx == -1: return 'L'
    elif dy == 1: return 'F'
    elif dy == -1: return 'B'
    return 'S'

def preprocess_image(image):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image_tensor = tf.convert_to_tensor(image)
    return tf.expand_dims(image_tensor, axis=0)

start = (5, 5)
goal = (5, 0)

while True:
    ret, frame = cap.read()
    if not ret:
        print("Failed to grab frame.")
        break

    input_tensor = preprocess_image(frame)
    detections = detect_fn(input_tensor)

    boxes = detections['detection_boxes'][0].numpy()
    classes = detections['detection_classes'][0].numpy().astype(np.int32)
    scores = detections['detection_scores'][0].numpy()
    im_height, im_width, _ = frame.shape

    obstacles = []
    for i in range(min(len(scores), 50)):
        if scores[i] > 0.5:
            class_id = classes[i]
            if class_id not in LABELS:
                continue
            box = boxes[i]
            y1, x1, y2, x2 = box
            (left, top, right, bottom) = (
                int(x1 * im_width), int(y1 * im_height),
                int(x2 * im_width), int(y2 * im_height)
            )
            cx, cy = (left + right) // 2, (top + bottom) // 2
            gx, gy = cx * GRID_SIZE // im_width, cy * GRID_SIZE // im_height
            obstacles.append((gx, gy))
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
            cv2.putText(frame, LABELS[class_id], (left, top - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    grid = create_grid(obstacles)
    path = a_star(start, goal, grid)
    command = get_command_from_path(path)

    print("Path:", path)
    print("Command:", command)

    arduino.write(command.encode())

    cv2.imshow("ESP32-CAM Detection + Dynamic A*", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
```

```
arduino.close()
cv2.destroyAllWindows()
```