SMART HELMET ENFORCEMENT SYSTEM
USING YOLOv8 AND IOT

Complete Code With Execution Steps

Step 1: Gather Components:

- Raspberry Pi (with necessary peripherals like keyboard, mouse, and monitor)
- Pi Camera module
- Relay module
- Indication LED lights
- External power source (such as a USB power bank or a wall adapter)
- Switch
- Helmet detection dataset (images/videos of people wearing and not wearing helmets)

Step 2: Set Up Raspberry Pi:

- Install the Raspberry Pi OS (e.g., Raspbian) onto an SD card using a computer.
- Insert the SD card into the Raspberry Pi and connect peripherals (keyboard, mouse, monitor).
- Boot up the Raspberry Pi and follow the on-screen instructions to complete the setup process, including connecting to Wi-Fi.

Step 3: Install Dependencies:

- Open a terminal on the Raspberry Pi.
- Install necessary software dependencies such as Python, TensorFlow, OpenCV, and any other libraries required for YOLO and camera operations.

Step 4: Develop or Obtain YOLO Detection Model:

- Develop a YOLO-based object detection model for detecting helmets.
- Preprocess the dataset and annotate the dataset.
- Train the model using the helmet detection dataset to distinguish between helmet-wearing and non-helmet-wearing individuals.
- Validate and fine-tune the model to improve accuracy.
- Save the trained model as “best.pt”

Step 5: Transfer Model to Raspberry Pi:

- Connect the Raspberry Pi to the internet via Wi-Fi or Ethernet.
- Transfer the trained YOLO model file (e.g., weights and .cfg files) to the Raspberry Pi using SCP (Secure Copy Protocol) or other file transfer methods.
- Place the model files in the appropriate directory on the Raspberry Pi.

Step 6: Write Detection Code:

- Develop Python code to interface with the Pi Camera module and perform real-time helmet detection using the YOLO model.
- Integrate logic for controlling the relay module based on the detection results (e.g., enabling/disabling the bike's ignition).
- Implement code for handling LED indication lights and switch input.

Step 7: Connect Hardware Components:

- Attach the Pi Camera module to the Raspberry Pi's camera port.
- Connect the relay module to the Raspberry Pi's GPIO pins for controlling the bike's ignition system.
- Wire up the indication LED lights to GPIO pins for visual feedback.
- Optionally, connect a switch to GPIO pins for manual control or system activation.

Step 8: Test Hardware and Software Integration:

- Power on the Raspberry Pi and ensure all components are functioning correctly.
- Run the detection code and verify that the Pi Camera captures video feed and the YOLO model detects helmets accurately.
- Test the relay module to ensure it can control the bike's ignition system based on detection results.
- Check the indication LED lights for proper functionality.

Step 9: Finalize Installation and Mounting:

- Mount the Raspberry Pi and other components securely on the motorcycle.
- Ensure all connections are stable and insulated against environmental factors (vibration, moisture, etc.).

Step 10: Field Testing and Calibration:

- Conduct field testing to validate the system's performance under real-world conditions.
- Fine-tune detection parameters and system behavior as necessary to optimize performance and reliability.

Step 11: Deployment and Maintenance:

- Deploy the Intelligent Helmet Enhancement System on motorcycles to enhance rider safety.
- Regularly inspect and maintain the system components to ensure proper functioning and reliability.
- Monitor system performance and address any issues or updates as needed.

Training Code for Helmet Object Detection Using YOLOv8

```
pip install Ultralytics

from ultralytics import YOLO
model = "yolov8l.pt"
data = "dataset yaml file location"
epochs = 100
imgsz = 640

yolo = YOLO()

yolo.train(
   task="detect",
   mode="train",
   model=model,
```

```
    data=data,
    epochs=epochs,
    imgsz=imgsz,
)
```

Detection Code Using the YOLOv8 Pretrained Model

```
from ultralytics import YOLO
import cv2
import cvzone
import math

cap = cv2.VideoCapture(0)  # For Video

model = YOLO("best(3).pt")

classNames = ['With_helmet', 'Without_helmet']
myColor = (0, 0, 255)
while True:
    success, img = cap.read()
    results = model(img, stream=True)
    for r in results:
        boxes = r.boxes
        for box in boxes:
            # Bounding Box
            x1, y1, x2, y2 = box.xyxy[0]
            x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
            # cv2.rectangle(img,(x1,y1),(x2,y2),(255,0,255),3)
            w, h = x2 - x1, y2 - y1
            # cvzone.cornerRect(img, (x1, y1, w, h))

            # Confidence
            conf = math.ceil((box.conf[0] * 100)) / 100

            cls = int(box.cls[0])
            currentClass = classNames[cls]
            print(currentClass)
            if conf > 0.5:
                if currentClass == 'Without_helmet':
                    myColor = (0, 0, 255)
                elif currentClass == 'With_helmet':
                    myColor = (0, 255, 0)
                else:
                    myColor = (255, 0, 0)

                cvzone.putTextRect(img, f'{classNames[cls]} {conf}',(max(0, x1), max(35, y1)), scale=1,
thickness=1, colorB=myColor, colorT=(255, 255, 255), colorR=myColor, offset=5)

                cv2.rectangle(img, (x1, y1), (x2, y2), myColor, 3)

    cv2.imshow("Image", img)
    cv2.waitKey(1)
```

Python Code to detect Helmet and Control the IOT Components

```
from flask import Flask, render_template, Response
import cv2
from ultralytics import YOLO
from picamera2 import Picamera2
from rpi_lcd import LCD
import RPi.GPIO as GPIO
from time import sleep
import time
import subprocess

def check_wifi_connection():
    try:
        result = subprocess.run(['iwconfig'], capture_output=True, text=True)
        if 'ESSID' in result.stdout:
            return True
        else:
            return False
    except Exception as e:
        print(f"Error checking WiFi connection: {e}")
        return False

def get_ip_address():
    # Get the IP address of wlan0 interface
    try:
        ip_address = subprocess.check_output(['hostname', '-I']).decode().strip()
        return ip_address
    except subprocess.CalledProcessError:
        return None

Red_led = 25
Green_led = 10
Motor = 24
But = 18
Buz = 23

detectedName = ''
status = 0
tim = 0
d = 0

GPIO.setwarnings(False)
lcd = LCD()
GPIO.setmode(GPIO.BCM)

GPIO.setup(Red_led, GPIO.OUT)
GPIO.setup(Green_led, GPIO.OUT)
GPIO.setup(Motor, GPIO.OUT)
GPIO.setup(But, GPIO.IN)
GPIO.setup(Buz, GPIO.OUT)
model = YOLO(r'best.pt')

lcd.text('Smart Bike', 1)
```

```
lcd.text("Connecting...    ", 2)
while 1:
    if check_wifi_connection():
        break
lcd.text("Fetching IP  ......", 2)
sleep(2)
ip = get_ip_address()
lcd.text(ip, 2)

picam2 = Picamera2()
picam2.preview_configuration.main.size = (640,480)
picam2.preview_configuration.main.format = "RGB888"
picam2.preview_configuration.align()
picam2.configure("preview")
picam2.start()

def detect(image):
    results = model(image,verbose=False)[0]
    Count = len(results)
    annotated_frame = results.plot()
    class_names = ''
    if Count > 0:
        class_names = [results.names[int(class_id)] for class_id in results.boxes.cls][0]
    return annotated_frame,class_names

app = Flask(__name__)
def gen_frames():
    global detectedName
    global status
    global tim
    global d
    global picam2
    while True:
        frame= picam2.capture_array()
        but = GPIO.input(But)
        frame,detectedName = detect(frame)
        if but == 1 and detectedName == 'helmet':
            GPIO.output(Buz, GPIO.HIGH)
            GPIO.output(Red_led, GPIO.LOW)
            GPIO.output(Green_led, GPIO.HIGH)
            GPIO.output(Motor, GPIO.HIGH)
            sleep(0.2)
            GPIO.output(Buz, GPIO.LOW)
            lcd.text('Bike started   ', 2)
            sleep(0.5)
            status = 1
        else:
            if status == 1 and but == 1:
                GPIO.output(Red_led, GPIO.LOW)
                GPIO.output(Green_led, GPIO.LOW)
                status = 0
                GPIO.output(Motor, GPIO.LOW)
                lcd.text(f'                  ', 2)
                d = 0
        if status == 1 and detectedName == 'head' and d ==0:
```

```
            GPIO.output(Red_led, GPIO.HIGH)
            GPIO.output(Green_led, GPIO.LOW)
            tim = time.time()
            d =1
            sleep(0.2)
            GPIO.output(Buz, GPIO.LOW)
        if status == 1 and detectedName == 'helmet':
            d = 0
            GPIO.output(Red_led, GPIO.LOW)
            GPIO.output(Green_led, GPIO.HIGH)
            lcd.text(f'                ', 2)
        if d == 1:
            l_time = int(time.time() - tim)
            #print(l_time)
            lcd.text('Time : '+str(10 - l_time), 2)
            if l_time > 10:
                status = 0
                d = 0
                GPIO.output(Red_led, GPIO.LOW)
                GPIO.output(Green_led, GPIO.LOW)
                GPIO.output(Motor, GPIO.LOW)
                lcd.text('Bike Stoped   ', 2)
                GPIO.output(Buz, GPIO.HIGH)
                sleep(0.5)
                lcd.text(f'                ', 2)
                GPIO.output(Buz, GPIO.LOW)

        ret, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == "__main__":
    app.run(debug=False,host='0.0.0.0')
```