PHASE -3 DEVELOPMENT

Objective:

The objective of this project is to create a traffic management system using Python and Raspberry Pi, which includes traffic light control, pedestrian crossing signals, ambulance detection, and real-time feedback on an LCD display. This system aims to efficiently manage traffic, improve pedestrian safety, and respond to emergency vehicles.

Implementation:

The traffic management system is implemented using Python on a Raspberry Pi. It integrates various hardware components such as LED lights, ultrasonic sensors, sound sensors, and an I2C LCD display. These components are controlled to manage traffic and provide visual feedback.

Components:

- **LED Traffic Lights**: Red, yellow, and green LED lights are used to control traffic signals for vehicles.
- **Pedestrian Crossing Signals**: Additional LEDs are employed to indicate pedestrian crossing signals, including red and green lights.
- **Ultrasonic Sensor:** The ultrasonic sensor is used to detect traffic density and trigger traffic light changes based on congestion.
- **Sound Sensor:** This sensor detects ambulance sirens, alerting the system to prioritize the passage of emergency vehicles.
- **I2C LCD Display:** An I2C LCD is used to provide real-time information about traffic conditions and signal states.

Steps:

- **Traffic Light Control**:

The code controls the red, yellow, and green LED traffic lights to manage vehicle traffic.The state variable current_state tracks the current traffic state. The red_duration, yellow_duration, and green_duration variables specify the duration for each traffic light phase.

```
red_light = LED(17)
yellow_light = LED(27)
green_light = LED(22)
current_state = "low_traffic"
red_duration = 15
green_duration = 20
yellow_duration = 3
```

```
while True:
   if current_state == "low_traffic":
      red_light.on()
      yellow_light.off()
      green_light.off()
      time.sleep(red_duration)
   # Handle high traffic scenario by changing the state and lights.
```

❖ **Pedestrian Crossing Signals**:

Additional LEDs control pedestrian crossing signals.The code synchronizes pedestrian signals with traffic lights to ensure pedestrian safety.

```
pedestrian_red = LED(5)
pedestrian_green = LED(6)

# Inside the traffic light control loop
if current_state == "low_traffic":
   # Pedestrians wait
   pedestrian_red.on()
   pedestrian_green.off()
   # Handle pedestrian crossing when the state changes to "high_traffic."
```

❖ **Ambulance Detection:**

The system listens for ambulance sirens using the sound sensor.When an ambulance is detected, it activates the yellow traffic light and pedestrian red signal to allow the ambulance to pass.

```
ambulance_siren = InputDevice(23)

# Inside the traffic light control loop
if ambulance_siren.is_active:
   yellow_light.on()
   pedestrian_red.on()
   # Handle ambulance passage and reset the lights after a brief period.
```

❖ **LCD Display:**

An I2C LCD display provides real-time feedback on traffic conditions, signal states, and any emergency alerts.

```
from RPLCD.i2c import CharLCD

lcd = CharLCD('PCF8574', 0x27)
lcd.clear()

# Inside the traffic light control loop
lcd.cursor_pos = (1, 0)
lcd.write_string(f"Red: {red_light.is_active}")
lcd.cursor_pos = (1, 8)
lcd.write_string(f"Yellow: {yellow_light.is_active}")
lcd.cursor_pos = (2, 0)
lcd.write_string(f"Green: {green_light.is_active}")
```

CODE:

```
from gpiozero import LED, DistanceSensor, InputDevice
import time
from RPLCD.i2c import CharLCD  # Library for I2C LCD display

# Traffic light control
red_light = LED(17)
yellow_light = LED(27)
green_light = LED(22)

# Pedestrian crossing signals
pedestrian_red = LED(5)
pedestrian_green = LED(6)

# Ultrasonic sensor for traffic density
traffic_sensor = DistanceSensor(echo=4, trigger=18, max_distance=2)

# Sound sensor for ambulance detection
ambulance_siren = InputDevice(23)

# LCD display setup
lcd = CharLCD('PCF8574', 0x27)
lcd.clear()

# Traffic state variables
current_state = "low_traffic"
red_duration = 15  # Duration for red light in seconds
green_duration = 20  # Duration for green light in seconds
yellow_duration = 3  # Duration for yellow light in seconds

# Pedestrian crossing intervals
pedestrian_wait_duration = 3  # Time for pedestrians to wait
pedestrian_cross_duration = 5  # Time for pedestrians to cross

while True:
    # Traffic density detection
    if traffic_sensor.distance < 0.5:  # Adjust this threshold
        if current_state != "high_traffic":
            current_state = "high_traffic"
            green_light.on()
            yellow_light.off()
            red_light.off()
            pedestrian_red.on()
            pedestrian_green.off()
            lcd.clear()
            lcd.write_string("High Traffic")
            time.sleep(green_duration)
    else:
        if current_state != "low_traffic":
            current_state = "low_traffic"
            red_light.on()
            yellow_light.off()
            green_light.off()
            pedestrian_red.off()
            pedestrian_green.on()
```

```
        lcd.clear()
        lcd.write_string("Low Traffic")
        time.sleep(red_duration)

# Ambulance detection
if ambulance_siren.is_active:
    yellow_light.on()
    pedestrian_red.on()
    lcd.clear()
    lcd.write_string("Ambulance Alert")
    time.sleep(yellow_duration)  # Keep yellow light on for a short duration
    pedestrian_red.off()
    time.sleep(yellow_duration)  # Allow pedestrians to cross

# Traffic light status on LCD
lcd.cursor_pos = (1, 0)
lcd.write_string(f"Red: {red_light.is_active}")
lcd.cursor_pos = (1, 8)
lcd.write_string(f"Yellow: {yellow_light.is_active}")
lcd.cursor_pos = (2, 0)
lcd.write_string(f"Green: {green_light.is_active}")

# Pedestrian crossing logic
if pedestrian_green.is_active:
    time.sleep(pedestrian_cross_duration)
    pedestrian_green.off()
    pedestrian_red.on()
else:
    time.sleep(pedestrian_wait_duration)
    pedestrian_red.off()
    pedestrian_green.on()

red_light.on()
green_light.off()
time.sleep(1)  # Delay between light changes
red_light.off()
yellow_light.on()
time.sleep(1)
yellow_light.off()
```