

# Final Project Planning Guide

## 1. Project Overview

**Project Title:** Smart Adaptive Traffic Management System with Simulation & Live Mode

**Subject/Course:** Computer Science / IoT & Embedded Systems **Team Members:** [Insert Names] **Due**

**Date:** [Insert Date]

## 2. Problem Statement & Objective

*What specific problem are you solving, or what question are you answering?*

- **The Problem:** Traditional traffic signal systems rely on static timing, leading to unnecessary congestion, increased waiting times, fuel wastage, and poor handling of emergency vehicles.
- **The Goal:** To build an intelligent system that dynamically adjusts green signal timing based on real-time vehicle density (detected via camera), while offering a low-cost simulation mode (using Arduino + LEDs) for demonstration.
- **Target Audience:** Traffic Management Authorities, Smart City Planners, and Academic Evaluators (for the college demo).

## 3. Scope & Requirements

*Define the boundaries of the project to prevent "scope creep".*

### Must-Have (Core Deliverables)

- [ ] **Vehicle Detection Engine:** Python/OpenCV based counting using background subtraction from video feeds.
- [ ] **Dynamic Timing Logic:** Algorithm to allocate green time based on density, with a "5-second pre-lock" feature.
- [ ] **Admin Panel:** Web interface (Flask/HTML) showing 4-direction live feeds, vehicle counts, and signal status.
- [ ] **Simulation Mode:** Hybrid operation using 1 live camera + 3 prerecorded videos, interfacing with Arduino to control LEDs.
- [ ] **Hardware Integration:** Arduino UNO + 4-Channel Relay + LEDs responding to Python logic via Serial.

### Nice-to-Have (Stretch Goals)

- [ ] **Emergency Priority:** Detect ambulances/emergency vehicles to trigger immediate green signals.
- [ ] **Night/Low Traffic Mode:** Minimal CPU usage mode with fixed short timers for off-peak hours.
- [ ] **Manual Override:** "Traffic Police Mode" allowing admin to pause auto-mode and control signals manually.
- [ ] **Fail-Safe:** Auto-switch to static timing if camera input fails.

## 4. Methodology / Technology Stack

How will you build or research this?

- **Backend:** Python, Flask (Web Framework), OpenCV (Computer Vision), `pyserial` (Hardware Comm).
- **Frontend:** HTML5 (Jinja2 Templates), Tailwind CSS (via CDN), Vanilla JavaScript (Fetch API).
- **Hardware:** Arduino UNO, 4-Channel Relay Module, Traffic LEDs.
- **Data Handling:** In-memory processing (No database required to reduce latency).
- **Key Concepts:** Background Subtraction, Serial Communication, MJPEG Streaming.

## 5. Timeline & Milestones

Work backward from the due date.

Phase	Description	Deadline	Status
Phase 1	<b>Setup &amp; Research:</b> Environment setup (Python/Flask), basic OpenCV vehicle detection test.	MM/DD	<div></div>
Phase 2	<b>Core Logic:</b> Implement counting logic across 4 streams and the dynamic timing algorithm.	MM/DD	<div></div>
Phase 3	<b>Hardware Integration:</b> Connect Arduino, write C++ firmware, and establish Python-Arduino serial comms.	MM/DD	<div></div>
Phase 4	<b>Frontend &amp; Modes:</b> Build Admin Panel, implement Simulation vs. Normal mode switching.	MM/DD	<div></div>
Phase 5	<b>Optimization &amp; Testing:</b> Reduce CPU usage, test fail-safes, and final documentation.	MM/DD	<div></div>

## 6. Risk Assessment

What could go wrong?

- **Risk:** High CPU lag processing 4 video feeds. -> **Mitigation:** Use lightweight Background Subtraction instead of heavy deep learning models (YOLO); optimize frame resizing.
- **Risk:** Arduino serial disconnects. -> **Mitigation:** Implement a try-catch block in Python to reconnect automatically or switch to a "Software-Only" mode.
- **Risk:** Lighting changes affecting detection. -> **Mitigation:** Use adaptive thresholding in OpenCV and provide a sensitivity slider in the Admin Panel.

## 7. Next Steps (Action Items)

1. ☐ Install Python, Flask, and OpenCV; set up the project folder structure.
2. ☐ Write a simple script to read a video file and draw contours around moving objects.
3. ☐ Order or gather hardware components (Arduino, Relay, LEDs).