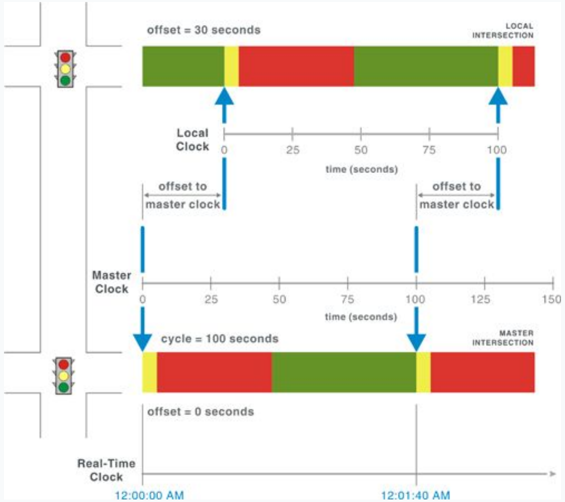
Design Review Presentation

D8 - Traffix

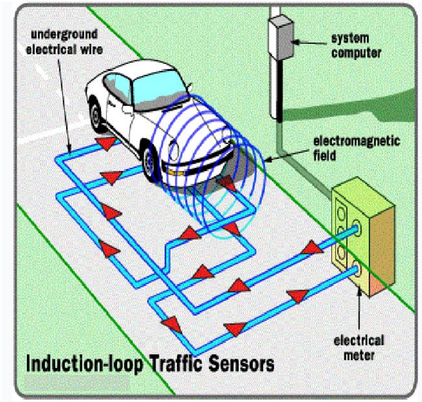
Use Case

The Problem:

- Current traffic lights **waste time and fuel**
- Stakeholders:
  - Local transportation authorities
  - Average commuter



[1]



[2]

The Solution:

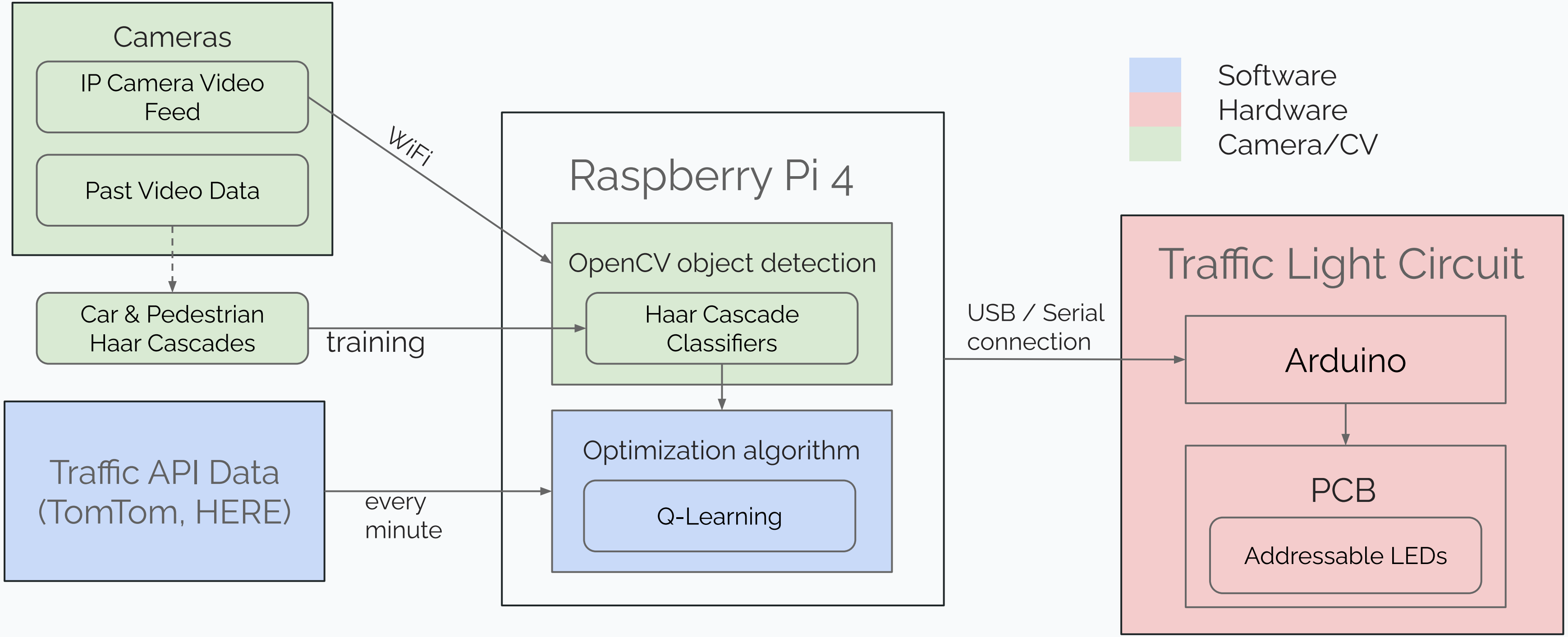
- Design a **smart traffic light that continuously optimizes light timings** based on car/pedestrian density and flow data

[1] https://ops.fhwa.dot.gov/publications/fhwahop08024/chapter6.htm
[2] https://www.researchgate.net/figure/Inductive-loop-detectors-based-traffic-management_fig1_274270897

Quantitative Design Requirements

Design Requirement	Specification	Use Case Justification
CV model accuracy	~90% for cars ~80% for pedestrians	Users should feel like light timings reflect actual traffic density
Optimization	Avg. wait time reduced >10% compared to fixed-time light	Q.O.L. improvement should be noticeable to drivers + pedestrians
Stress/complexity handling	Models can handle a minimum of 10 cars at each side of intersection + complex API data	Product is most useful if it can be used to alleviate high-density traffic
Latency	< 5s total between traffic data input and time interval update	Light changes should accurately reflect the current situation

System Specification



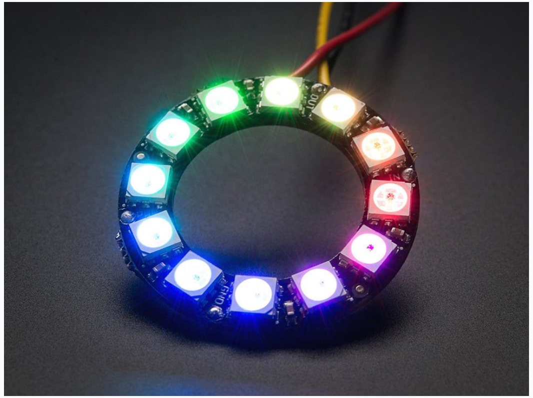
Solution Approach - Hardware

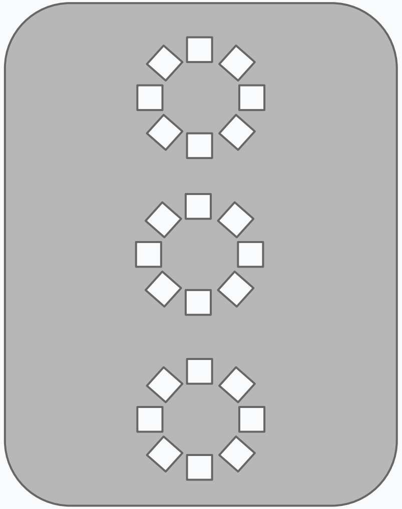
Cameras

- 4 IP cameras to capture each side of the intersection
- Live video data streamed to RPi (WiFi connection)

Traffic Light Mockup

- Addressable LED Ring
- Controlled by Arduino
  - USB connection to RPi





Implementation - Hardware

Traffic Light Circuit (TLC)

- Custom PCB: breakout board mounted on an Arduino
- Arduino fed light timing data from RPi
  - Translates data to control addressable LEDs

Camera Setup

- 4 WiFi enabled cameras, one for each street direction
- Camera positions must be fairly consistent for image identification model
  - Depending on testing process, may need to construct a mount of some sort

Reolink Argus 2E

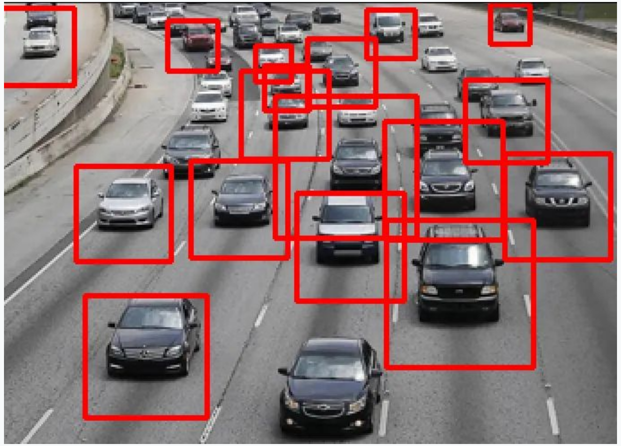
Solution Approach - Software

Traffic Object Identification Model

- Identify number of cars and number of pedestrians
- Feed into optimization model to determine how to change light intervals

Optimization Model

- Traffic APIs for live data of nearby intersections with oncoming traffic
- Data from object identification model to get current intersection data
- Reinforcement learning to allow historical data to influence future light intervals



https://medium.com/@kaanerdenn/introduction-to-object-detection-vehicle-detection-with-opencv-and-cascade-classifiers-8c6834191a0b

Implementation - Software

Optimization Model

- Live traffic data of nearby roads: TomTom Traffic API and HERE Traffic API
  - Free for our usage
- Reinforcement learning techniques: Q-learning[1]
- Pytorch

Object Detection Model

- Haar cascade - easy to implement on constrained hardware
- Train different cascades for different objects
  - Use existing XML files for cars and pedestrians[2]
- OpenCV

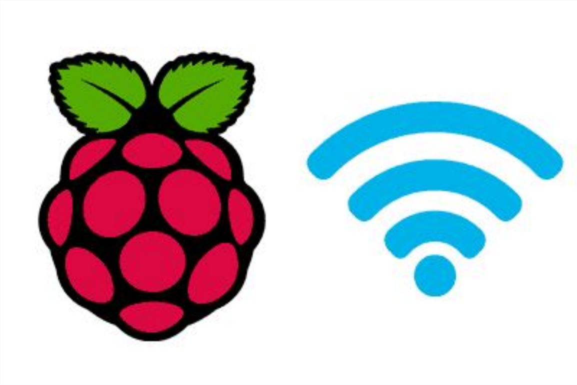

```
{
  "flowSegmentData": {
    "-xmlns": "http://lbs.tomtom.com/services",
    "-version": "traffic-service 2.0.004",
    "frc": "FRC2",
    "currentSpeed": 41,
    "freeFlowSpeed": 70,
    "currentTravelTime": 153,
    "freeFlowTravelTime": 90,
    "confidence": 0.59,
    "roadClosure": true,
    "coordinates": {
      "coordinate": [
        {
          "latitude": 52.40476,
          "longitude": 4.844318
        },
        {
          "latitude": 52.411312,
          "longitude": 4.8299975
        },
        {
          "latitude": 52.415073,
          "longitude": 4.827327
        }
      ]
```


[1] Based off of similar research: https://cs229.stanford.edu/proj2016spr/report/047.pdf
[2] Sourced by this repo: https://github.com/AdityaPai2398/Vehicle-And-Pedestrian-Detection-Using-Haar-Cascades/tree/master

Implementation - Integration

Processing/Computation

- Raspberry Pi 4 will run CV and optimization models
    - Connected to common WiFi network with cameras to receive their live data
        - CMU-SECURE or Mobile WiFi hotspot
    - Make API calls from RPi
    - Output light timing info sent through Serial communication to Arduino



[1]

[1] https://www.luisllamas.es/en/raspberry-pi-wifi/

Testing, Verification, Metrics

Optimization Model

- Compare average wait time of cars & pedestrians over multiple traffic cycles (2-5)
    - With simulated car and pedestrian counts using SUMO, against simulated fixed-interval model
    - With actual footage taken on each side of the intersection
    - > 10% reduction in average wait time

Object Detection Model

- Run on video samples and verify correct counts are achieved
    - ~90% accuracy with vehicles
    - ~80% accuracy with pedestrians

Testing, Verification, Metrics

Traffic Light Circuit (TLC)

- Integration tests to ensure:
  - Input RPi data is received properly
  - Output to LEDs reflects desired functionality
- RPi data receipt -> LED change latency should be < 1s

RPi Integration

- Test WiFi connection with cameras and ability to receive API call data
- Stress tests to verify latency < 5s between input and output to TLC
  - “stress” = high-complexity data, rather than high-speed

Risk Mitigation

Cameras

- Reduce initial 4-camera plan to 2 cameras (simulate other sides based on API data)
- Use standard wired RPi cameras if IP cameras fail
- Use pre-recorded videos / existing traffic camera footage if image identification model does not work reliably with live camera feed

Software

- If latency requirements not met
  - Only keep track of wait time for specific cars
  - Only run image identification model on 2 sides of the intersection
  - Only consider vehicles for optimization algorithm

Schedule (link)