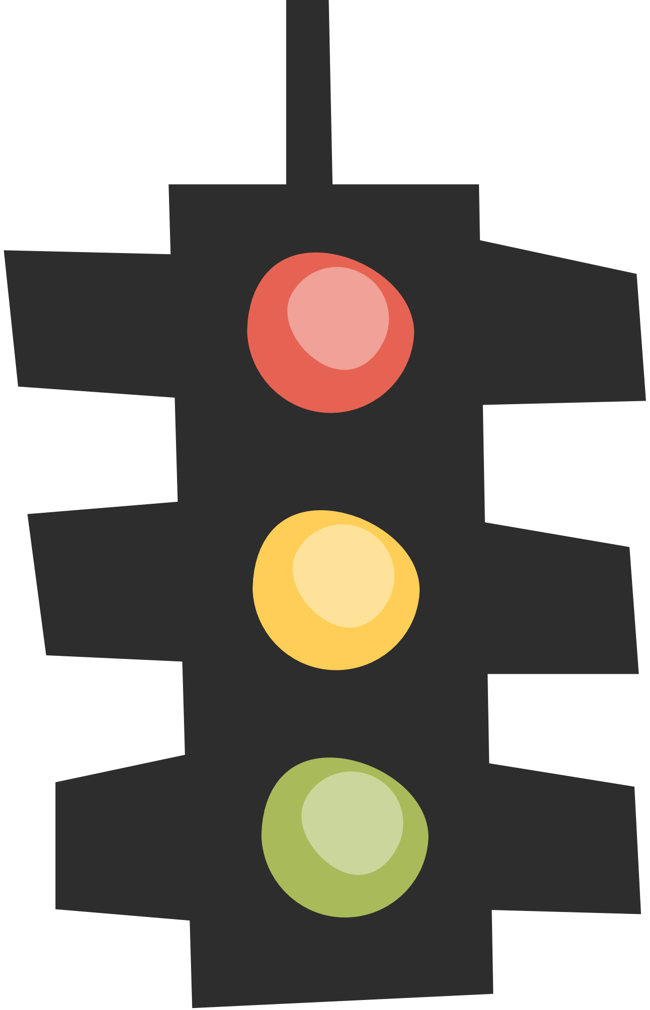
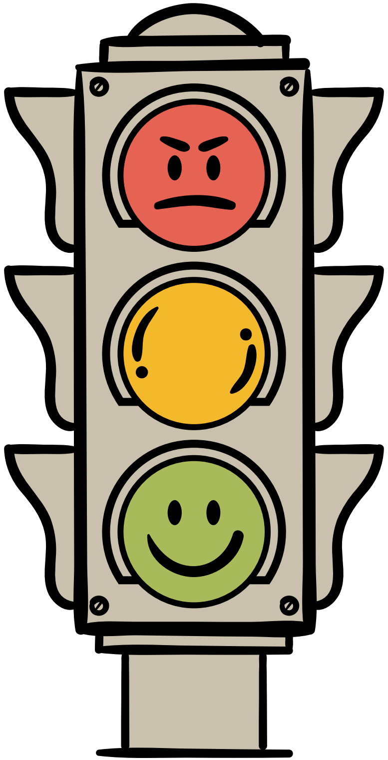


Team D8

Traffix

Ankita Chatterjee, Kaitlyn Liu, Zina Zarzycki

Use Case



THE PROBLEM

Current traffic lights **waste time and fuel** because they are not optimized for varying traffic conditions

Existing technologies like induction sensors **don't adapt** to evolving traffic patterns

STAKEHOLDERS

Local transportation authorities **save on long term costs** to optimize traffic

Average commuter saves on time wasted while commuting

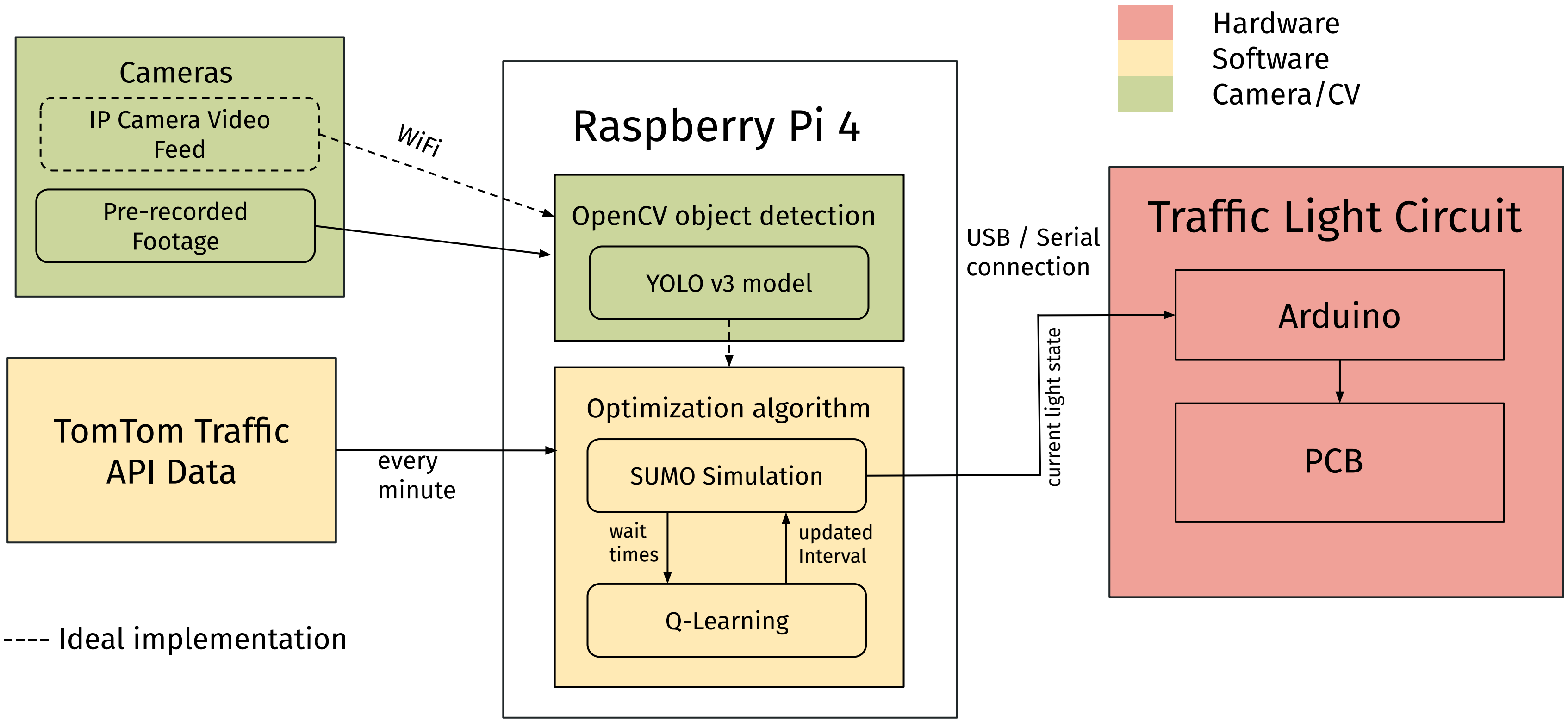
OUR SOLUTION

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

Replacement to existing traffic lights

Can be implemented in isolation or at city-wide level

System Specification



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

Implementation - Object Detection

Overall Solution

- Run on 4 concurrent videos from each side of Fifth & Craig intersection
- Detect number of pedestrians and cars in each frame with YOLOv3 model
- Determine lane boundaries in order to output number of cars and pedestrians on each side of the intersection
    - Currently using hard-coded coordinates as opposed to an edge detection algorithm

Demo Details

- Object detection code will run on pre-recorded footage
- Display vehicle and pedestrian counts for each side on monitor

Key Changes & Tradeoffs

Using YOLOv3 model instead of cascade classifiers

- Haar cascade accuracy was very low <50% due to false positives
- YOLOv4 model provided similar accuracy to YOLOv3 but higher latency

Using pre-recorded footage instead of a live camera feed for demo purposes

- Using wired IP cameras (powered with portable batteries) due to inability to access live stream of battery-powered IP cameras

Implementation - Optimization

Overall Solution

- Deep Q-learning model with Pytorch
  - 2 layered neural network
  - Huber loss function
- Toggleable online or offline model
- Called in TraCI script code to constantly update SUMO simulation traffic lights
- Outputs (North-South Green duration, East-West Green duration) to the simulation
- State input:
  - queue length, average speed, current light phase, time left in phase

Demo Details

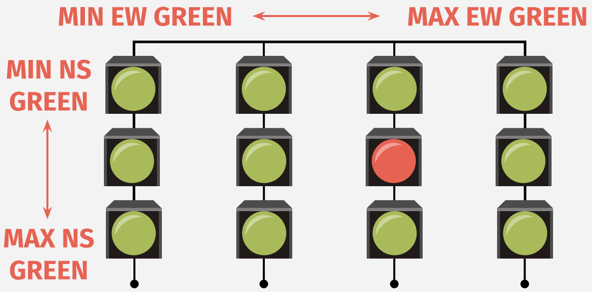
- Using simulated pedestrian and vehicle counts during demo instead of camera data input
- Vehicles in footage will not respond to simulated light changes leading to optimization not working

Key Changes

Light interval instead of color action states

- (North-South green duration, East-West green duration) vs North-South at single time/interval
- Safety - delayed updates don't harm upcoming cycles
- Easier to implement - no need for external timing mechanism

Action representation:



Implementation - Simulation

Overall Solution

- Using SUMO traffic simulator w/ TraCI Python
- Polled constantly by traffic light circuit to determine current state of physical traffic light
- Lane area detectors to mimic object detection model
- Calibrators to simulate real life traffic flow from TomTom API

Demo Details

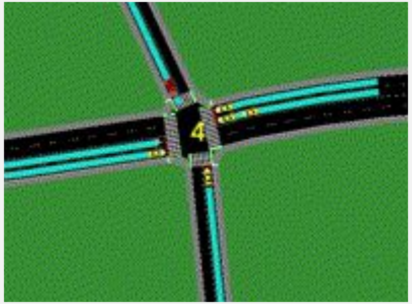
- Plans to implement 3D modeled simulation for demo
- Will also output live state data
    - Cars at each side of intersection, average wait time, etc

Key Changes

- Only using TomTom API instead of TomTom and HERE
    - Redundant flow information

Example Simulation Feed

```
0 EW green
1 EW yellow
2 NS green
3 NS yellow
4 Pedestrian
```



Implementation - Circuit

Overall Solution

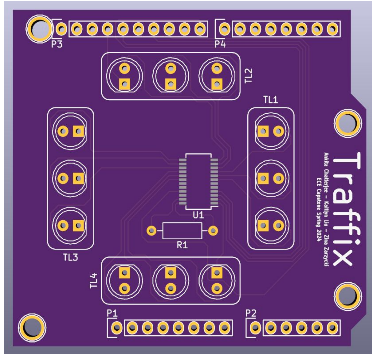
- RPi outputs current light state information, sends to Arduino using serial communication
- Arduino uses SPI transmission to update light ON/OFF states stored in the TLC5928 LED Driver chip
- LED Driver outputs are connected to 12 LEDs that model a four-way intersection
- Packaged together as a custom Arduino shield PCB

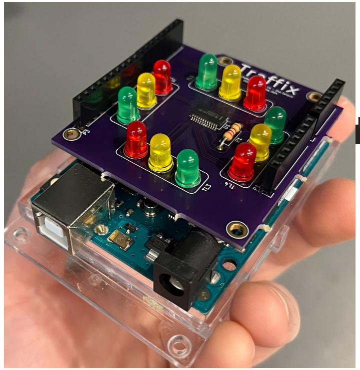
Demo Details

- The traffic light circuit will be connected to the RPi output, reflecting the optimized light timing patterns

Key Changes

- Using an LED driver chip to control an array of individual LEDs, rather than using addressable LED strips





Testing, Verification, Metrics - Optimization

HOW WE TESTED

- Comparing average wait time of cars in SUMO simulation with ML model controlling light durations to same periods without using the ML model
    - Over 8 periods of 1hr in simulation time for both trials
- Latency: tested over 10 iterations of interval calculation

FURTHER IMPROVEMENTS

- Simulation currently does not have a lot of randomness and could be closer to real life environment
    - Improve before demo with more route variability

AVERAGE WAIT TIME

WITHOUT OPTIMIZATION: 30.87 s

WITH OPTIMIZATION: 15.78 s

48.90 %
TIME REDUCTION

0.1024 s
LATENCY

Testing, Verification, Metrics - Object Detection

HOW WE TESTED

- 100 frames of pre-recorded video at Fifth and Craig intersection; all metrics averaged over those frames
- Compared actual object counts to object vehicle counts
- Maximum vehicles detected on one side with full accuracy was 11

FURTHER IMPROVEMENTS

- Need to re-test latency when all 4 frames are being processed concurrently; will probably get worse
- Only tested with 3 sides of the intersection because that is the only stable footage we have as of now
- Used hard-coded lane boundaries - may test edge detection algorithm

80 %
VEHICLE COUNT ACCURACY

90 %
PEDESTRIAN COUNT ACCURACY

11
MAXIMUM VEHICLE COUNT

~ 4 s
LATENCY

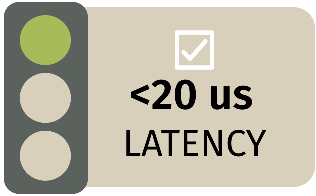
Testing, Verification, Metrics - Circuit

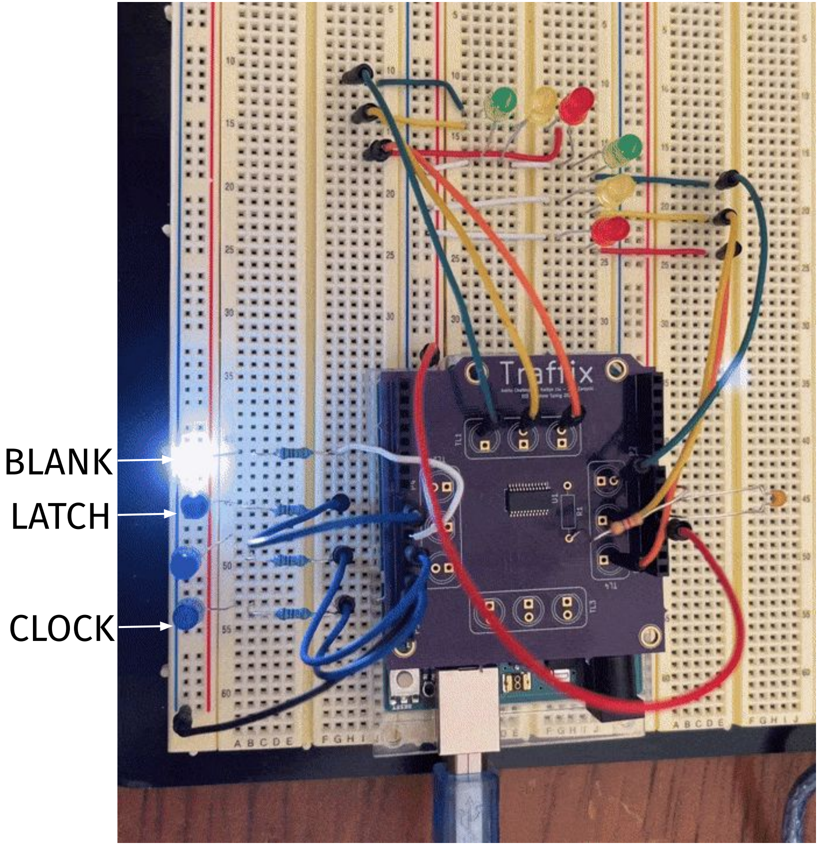
HOW WE TESTED

- Partially assembled one of the PCBs and wired it to breadboarded LEDs
- Ran Arduino TB to verify lights transition as intended
  - Discovered wiring issue with R_{IREF}
- Connected Arduino to RPi to verify serially-communicated control over light states
- Error statements printed to serial monitor allow us to ensure that no illegal light patterns happen

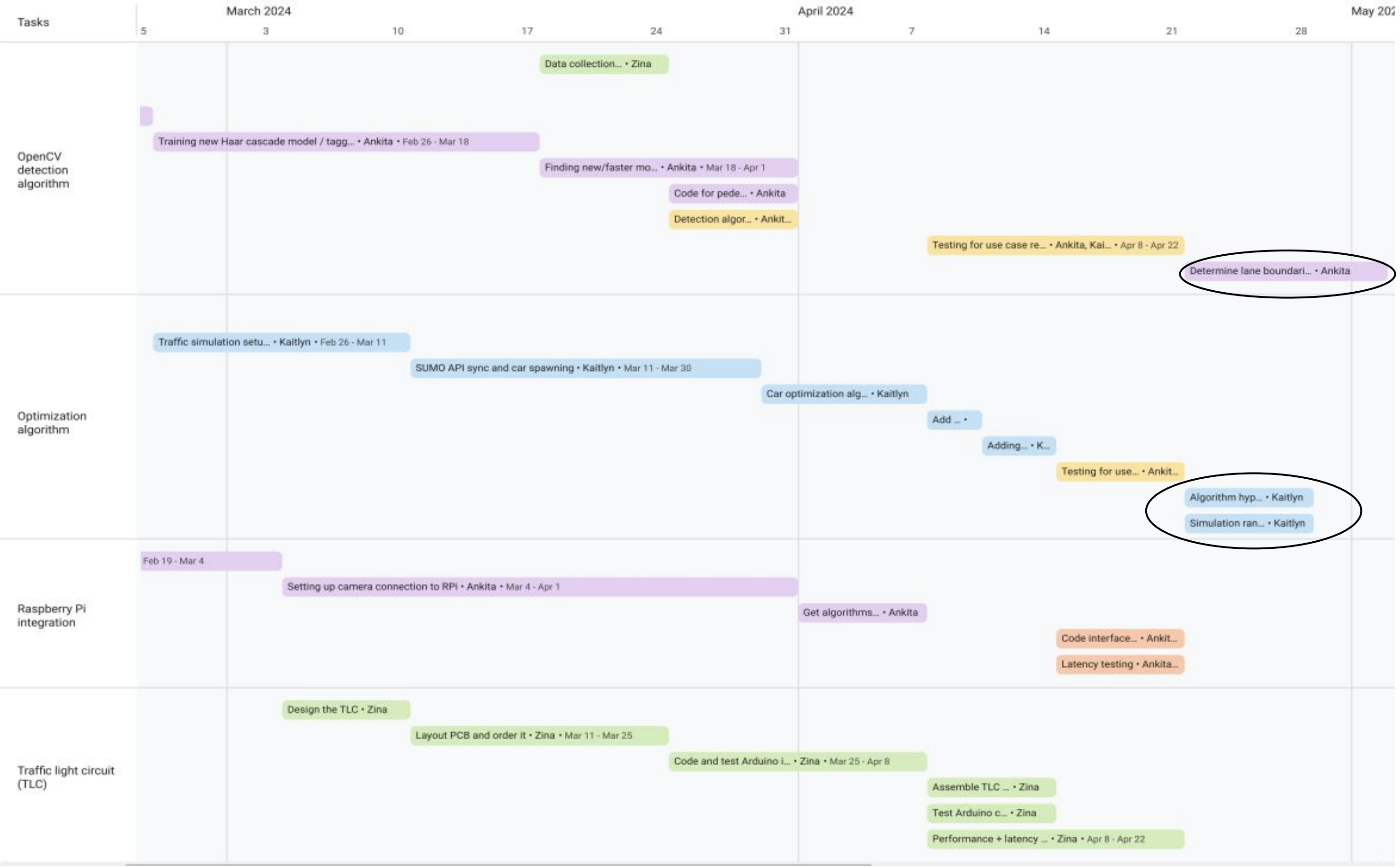
FURTHER IMPROVEMENTS

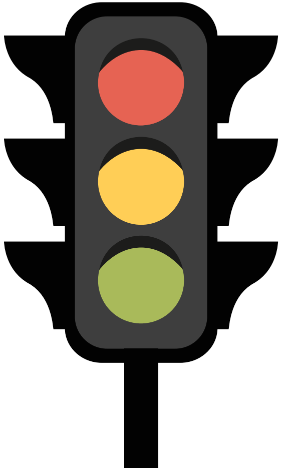
- Correctly wired PCBs have been ordered





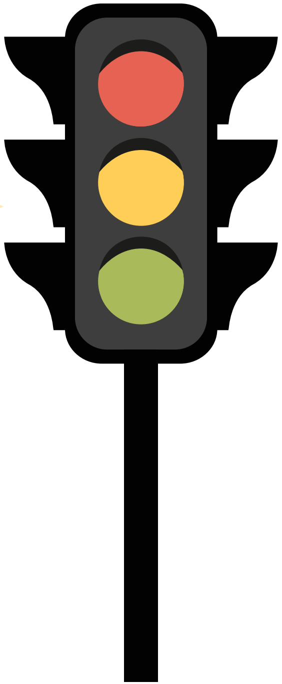
Schedule





*February tasks and deliverables not included, see website schedule for more details

Key Takeaways



TRY TO STICK TO WIDELY USED TOOLS/LIBRARIES

- Better documentation = more gentle learning curve
- More/quicker support

EVERYTHING TAKES LONGER THAN YOU THINK IT WILL

- Leave lots of slack time
- Check that things work ASAP

RESEARCH WHAT YOU ORDER

- Double and triple check that the parts you order are capable of doing what you need them to do to avoid setbacks and unnecessary purchases