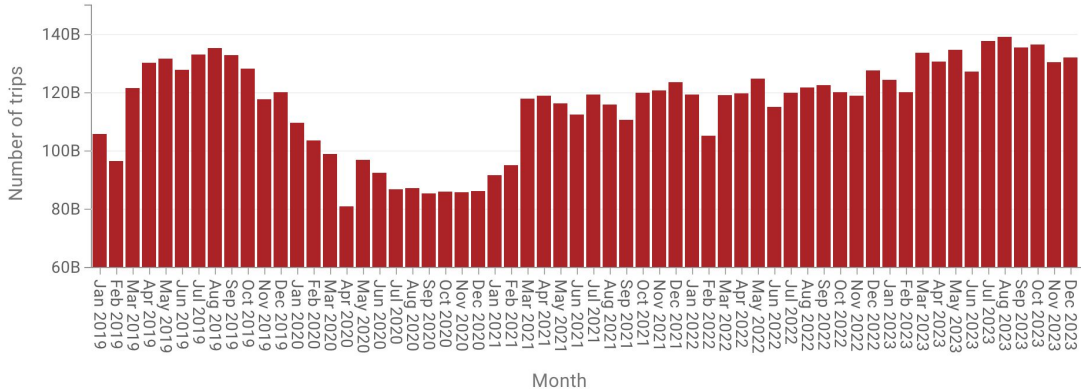
Team D8: Traffix

Ankita Chatterjee, Kaitlyn Liu, Zina Zarzycki

Use Case

- Americans drive **a lot**
- Current traffic light implementations are:
  - **Inefficient**
  - **Environmentally unfriendly**
  - **Unsafe**

Total number of car trips taken by Americans per month [1]



- Our solution
  - Design a **smart traffic light that optimizes light timings in real time** based on car/pedestrian density and flow data obtained via cameras + traffic API calls
  - Requires hardware, software, and circuits knowledge

[1] Data sourced from the USDoT's Bureau of Transportation Statistics (found at https://data.bts.gov/Research-and-Statistics/Trips-by-Distance/w96p-f2qv/about_data)

Use Case Requirements

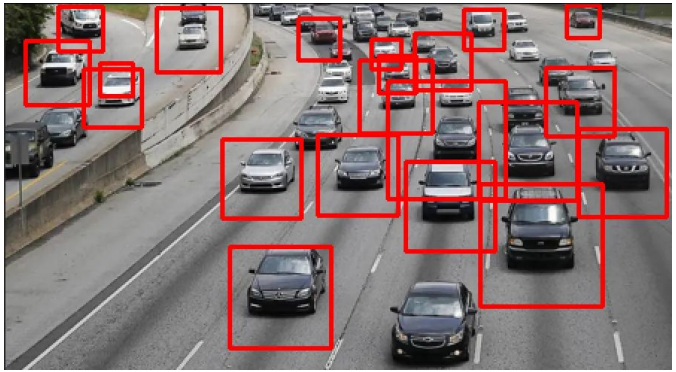
- Efficiency improvement
  - Average wait time reduction >10%
- Safety
  - Abides by traffic guidelines
- High traffic situations
  - Can handle >20 cars waiting at each side of the intersection

Use Case Requirements

- Can handle up to a four-way intersection of two-lane streets
  - Think Fifth & Craig
- Can default to a standard fixed-time light protocol

Technical Challenges

- Differentiating between relevant and irrelevant objects
- Optimizing light interval timings
- Syncing all lights at an intersection
- Real-world testing



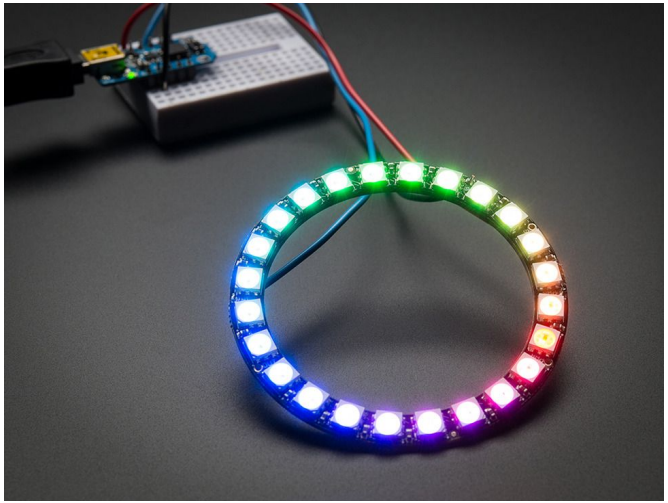
https://medium.com/@kaanerdenn/introduction-to-object-detection-vehicle-detection-with-opencv-and-cascade-classifiers-8c6834191a0b

Solution Approach - Hardware

- 4 IP Cameras:
  - Reolink Argus Eco
  - Battery-powered
  - Take pictures/video of intersection
    - Fifth Ave and Craig St
  - Data sent to Raspberry Pi via WiFi connection

https://reolink.com/us/product/argus-eco/?attribute_pa_version=1-pack-3mp-white-solar-panel-white

- Traffic Light Prototype:
  - PCB (breakout board for an Arduino or other microcontroller)
  - Addressable LEDs (Neopixels)



https://www.adafruit.com/product/1586

Solution Approach - Software

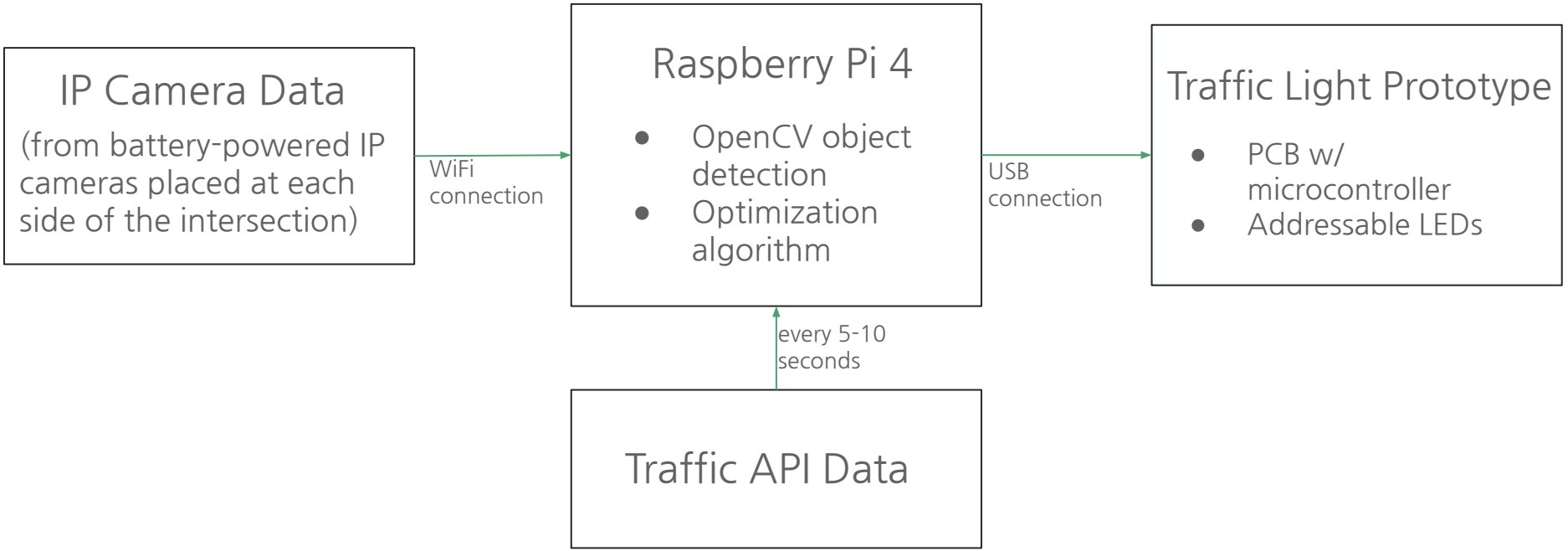
- Program to extract relevant traffic object counts
  - Current state data sourced from **camera feeds**
    - Haar feature-based cascade classifiers[1]
  - Future state data sourced from **traffic API**

- Traffic flow optimization algorithm
  - Local descent algorithm[2]
  - Optimize green light intervals to minimize avg. time spent waiting by all cars in one cycle

"Investigating Advanced Traffic Signal Control", California Department of Transportation
https://www.fhwa.dot.gov/publications/research/ear/11044/index.cfm

[1] Cascade classifiers: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
[2] Line search methods: https://optimization.cbe.cornell.edu/index.php?title=Line_search_methods

Solution Approach



Testing Strategy

- Traffic Object Detection Algorithm
  - Collect video samples and verify correct counts are achieved
- Optimization Algorithm
  - Simulate variety of randomized traffic scenarios
- Traffic Light Prototype
  - Test that PCB functionality matches that of breadboarded circuit

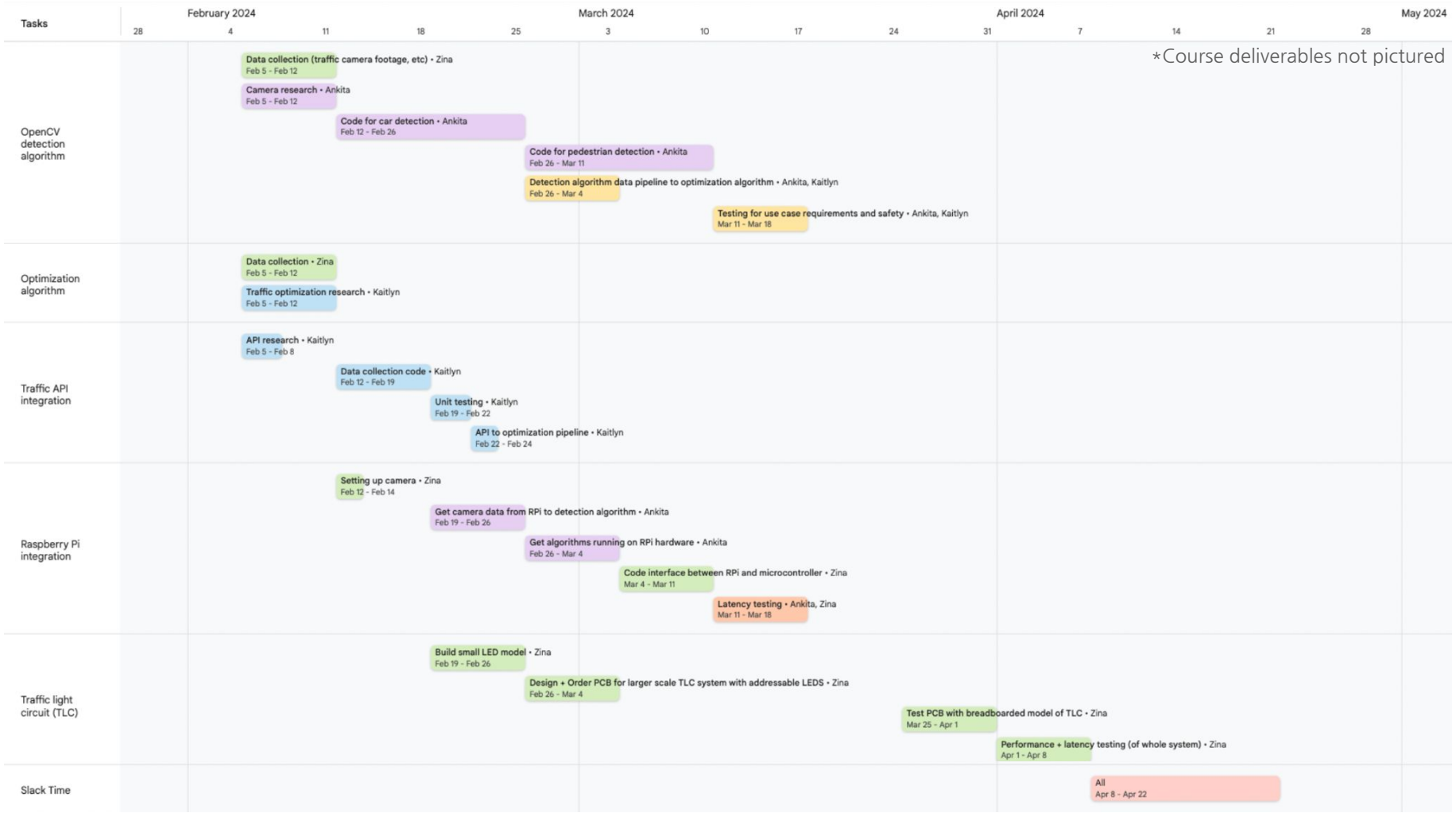
Verification Metrics

- Object detection accuracy
  - Car detection accuracy > 90% [1]
  - Pedestrian detection accuracy > 80% [2]
- Optimization
  - >10% average wait time reduction compared to existing protocols
- Latency
  - Expected light change delay < 3 sec.
  - Traffic object counts update every 5 sec.
  - Traffic API and camera data offset < 5 seconds

[1] Ahmad, A.B.; Tsuji, T. Traffic Monitoring System Based on Deep Learning and Seismometer Data. *Appl. Sci.* **2021**, *11*, 4590. https://doi.org/10.3390/app11104590
[2] C. Duan and S. Luo, "Design of Pedestrian Detection System based on OpenCV," 2022 4th International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM), Hamburg, Germany, 2022, pp. 256-259, doi: 10.1109/AIAM57466.2022.00055.

Division of Labor

- Data collection
  - Zina
- OpenCV car-counting algorithm
  - Ankita, Kaitlyn
- Traffic API data integration
  - Kaitlyn
- Light timing optimization algorithm
  - Ankita, Kaitlyn
- Raspberry Pi integration
  - Ankita, Zina
- Traffic light prototype design and implementation
  - Zina


Tasks
February 2024
March 2024
April 2024
May 2024
28
4
11
18
25
3
10
17
24
31
7
14
21
28
*Course deliverables not pictured
OpenCV detection algorithm
Data collection (traffic camera footage, etc) • Zina
Feb 5 - Feb 12
Camera research • Ankita
Feb 5 - Feb 12
Code for car detection • Ankita
Feb 12 - Feb 26
Code for pedestrian detection • Ankita
Feb 26 - Mar 11
Detection algorithm data pipeline to optimization algorithm • Ankita, Kaitlyn
Feb 26 - Mar 4
Testing for use case requirements and safety • Ankita, Kaitlyn
Mar 11 - Mar 18
Optimization algorithm
Data collection • Zina
Feb 5 - Feb 12
Traffic optimization research • Kaitlyn
Feb 5 - Feb 12
Traffic API integration
API research • Kaitlyn
Feb 5 - Feb 8
Data collection code • Kaitlyn
Feb 12 - Feb 19
Unit testing • Kaitlyn
Feb 19 - Feb 22
API to optimization pipeline • Kaitlyn
Feb 22 - Feb 24
Raspberry Pi integration
Setting up camera • Zina
Feb 12 - Feb 14
Get camera data from RPi to detection algorithm • Ankita
Feb 19 - Feb 26
Get algorithms running on RPi hardware • Ankita
Feb 26 - Mar 4
Code interface between RPi and microcontroller • Zina
Mar 4 - Mar 11
Latency testing • Ankita, Zina
Mar 11 - Mar 18
Traffic light circuit (TLC)
Build small LED model • Zina
Feb 19 - Feb 26
Design + Order PCB for larger scale TLC system with addressable LEDS • Zina
Feb 26 - Mar 4
Test PCB with breadboarded model of TLC • Zina
Mar 25 - Apr 1
Performance + latency testing (of whole system) • Zina
Apr 1 - Apr 8
Slack Time
All
Apr 8 - Apr 22