Presentation video URL:

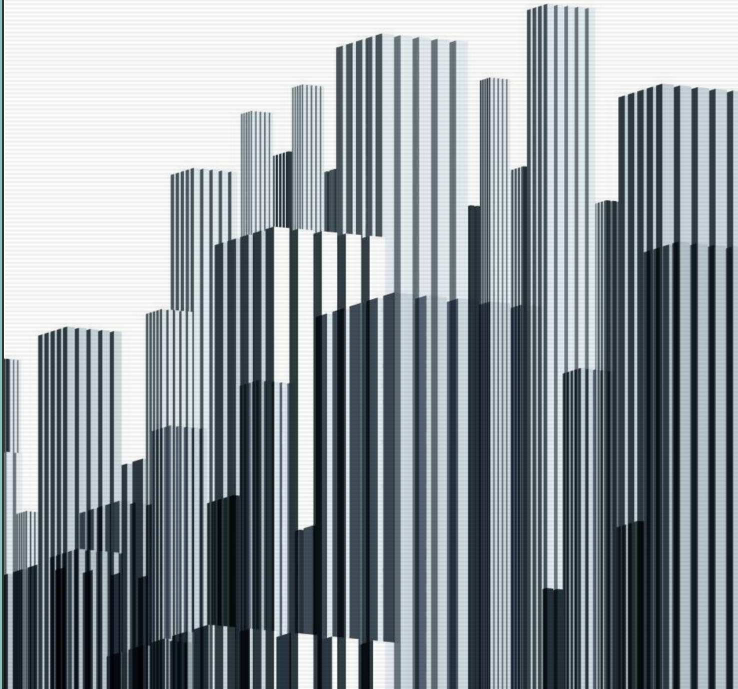
https://youtu.be/cr4ptFqgCUg

Apollo Conceptual Architecture Presented by - 40 Below

Team Leader - Joshua Kouri	•Subsystem breakdown, Subsystem dependencies, References
Presenter - Josh Tremblay	•Functional requirements, Presentation
Alex Nechyporenko	•Derivation, Non-functional requirements, Conclusion
Kennan Bays	•Abstract, Sequence diagrams
Mason Choi	•Intro, Architecture Overview, Lessons learned

Overview

- Derivation through analysis of functional & non-functional requirements. Additionally, describing alternatives for both.
- Overview of the architecture and the architectural style. Then, describing the functionalities of each subsystem and how they interact with each other.
- Details of limitations encountered during the assignment and the resulting lessons learned from them
- Concluding thoughts and summary of our findings



Functional Requirements

Perception
- Sense & gather data from surrounding environment
- Detect obstacles & traffic lights most importantly
- Process data to predict behaviours and status to react accordingly

Simulation
- Virtually drive autonomous vehicles
- Useful for testing, verifying, & optimizing system
- Gathers information for Apollo devs & familiarity for users

HD map & Localization
- Mapping road environment and related objects
- Determine location of vehicle for use of other functionalities

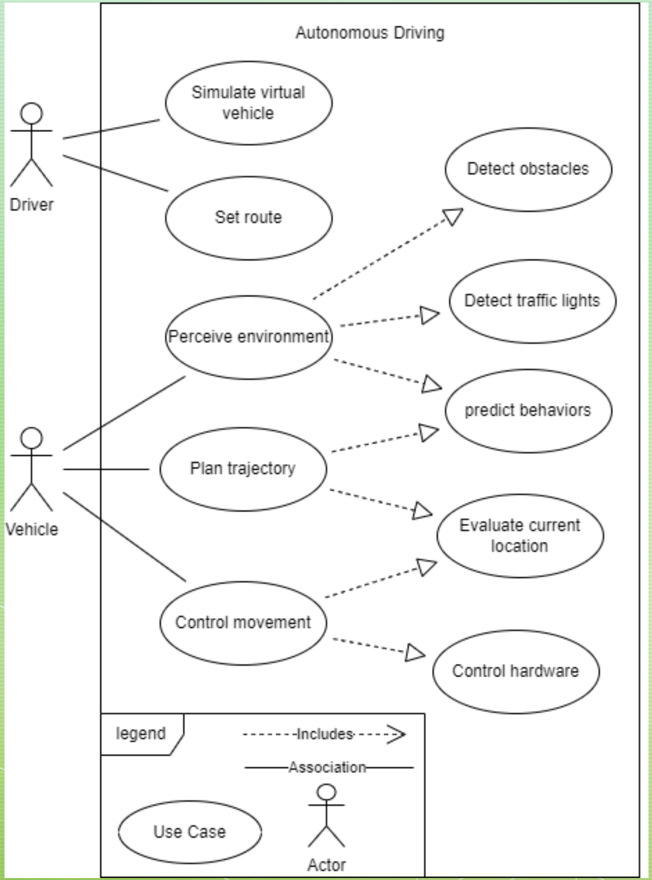
Functional Requirements (cont.) & Use Case Diagram

Control

- Performs the planned route
- Adapts to situations (such as road conditions, and speed limits)

Planning

- Plan a trajectory that the vehicle would perform
- Deciding on by predicted data, and behaviours gathered from perception



Alternative Derivation

General distribution of categories:

- Perception (Perception)
  - Data detected of surroundings and the interpretation of it
- Decision & Control (HD map, Localization, Planning)
  - Defining the way the vehicle can move based on route & location, environmental events, & state of system
- Vehicle Platform Manipulation (Control)
  - Allowing the vehicle to move based on Decision & Control
- What about Simulation?
  - Not applicable to above categories
  - Makes Apollo distinct

Concurrency and Security

Concurrency

- In order to facilitate the advancement of driverless cars, Apollo created a framework called Cyber RT which is a centralized system that manages parallel computing, along with high concurrency task execution
- Given the data intensity of autonomous vehicles and the asynchronous execution of multiple parallel systems, the framework must support high throughput of concurrent data transferring

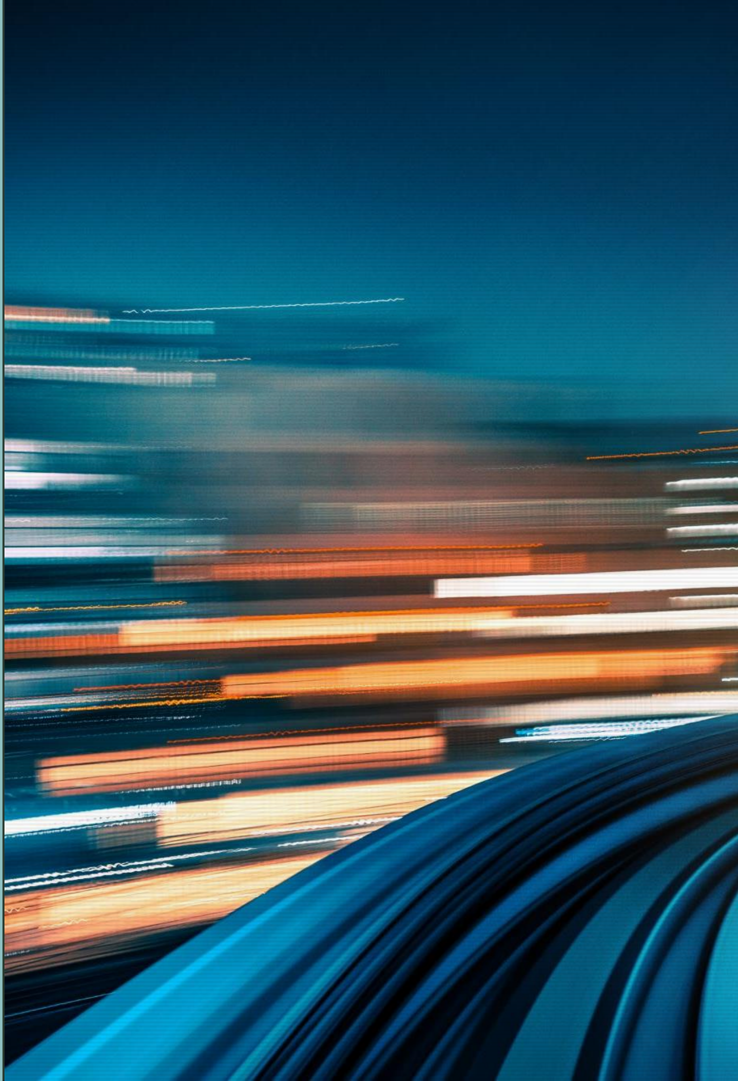
Security

- Due to autonomous vehicles not only impacting the security from physical harm of the passengers, but also other vehicles and bystanders, it is crucial that the software system is secure
- The consequences of security failures can be dire, therefore maintaining the integrity of the system is of essence
- Apollo protects from hackers access of the system, the data distribution and monitors any malicious instructions or software downloads

Alternative Quality - Portability

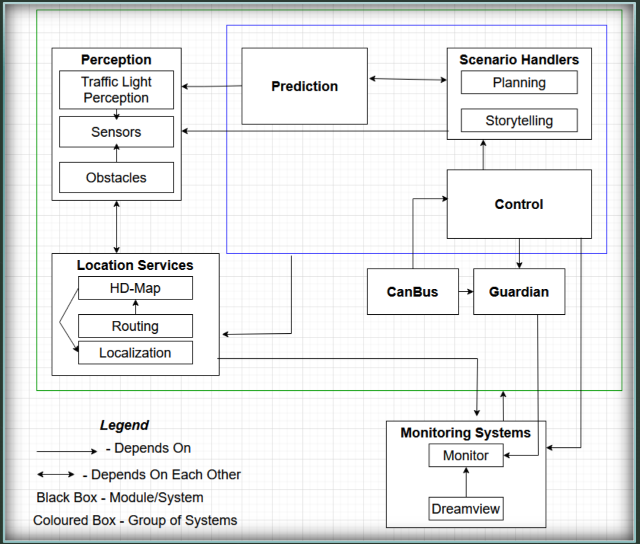
New Industries Perspective

- With the advent of driverless cars, the demand for new businesses that will depend on this technology will grow exponentially. Hence, Apollo should consider implementing the integration of the system with third-party applications in a secure fashion
- Companies such as Uber will inevitably start to rely on autonomous driving in order to capture that market Apollo can allow such applications to connect to the car and to give it instructions
- Giving Apollo the ability to have phones access the car, along with laptops may be the next step



Architecture Overview

- The primary module that the vehicle depends on is CanBus, which takes input from the control module and executes it.
- Overall, the architectural style used for Apollo appears to be a close match to process-control.
- Process-control architecture is often used for real-time system software such as those belonging to automobile cruise control, so it makes sense that Apollo is using it as well.
- A feedback control system is being used, as information about various process variables are used in order to adjust the output of the system.
- The controller in this system is the control module, and it obtains variables from the perception, prediction, and planning modules.



Subsystems

Here are the top-level overview of the subsystems utilized within the architecture:

- **Perception** – gathers information from environment (traffic-lights, obstacles and sensors**)** for other subsystems use.
- **Prediction** – takes information gathered by the Perception and Scenario Handler modules to predict the behavior of detected obstacles on the road.
- **Scenario Handlers** - pair of modules (planning & storytelling) which govern how the vehicle reacts the situation it is in.
- **Control** – takes inputs from the Scenario Handler modules which it processes and sends commands to the CanBus
- **Location Services** - understands where the vehicle is in relation to the world. The three submodules that accomplish this task are HD-Map, Localization, and Routing.
- **CanBus** - executes instructions from the control module, while simultaneously also collects data on the vehicle’s status to give feedback to the control module.
- **Monitoring Systems** - gathers information from all other modules to ensure that all systems are functioning properly

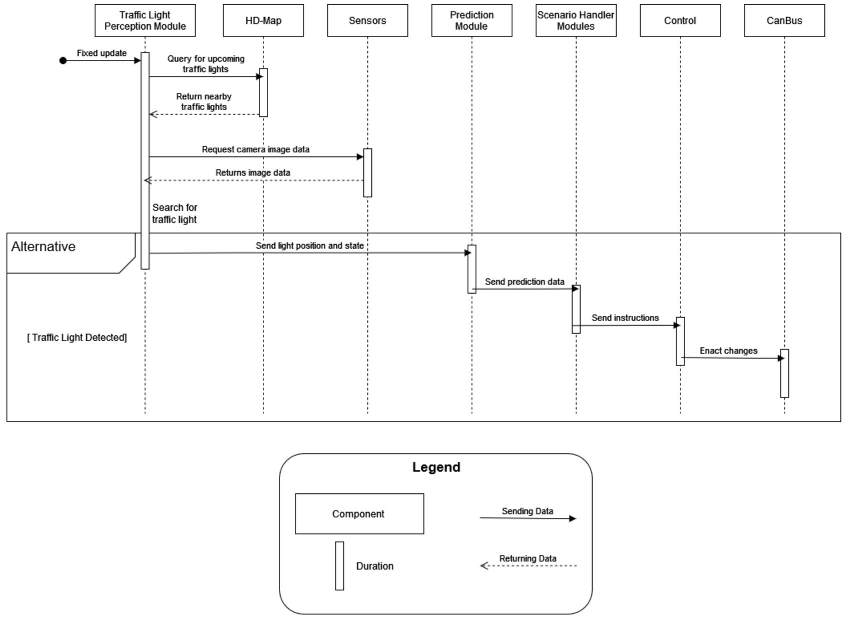
Subsystem Interactions

The subsystems can be thought of as the human system

- Perception - Eyes
  - Communicates information of the external world to the other components of the system
- Planning & Scenario Handlers – Brain
  - Creates decisions based on information that have been passed to it
- Control & CanBus – Nervous System
  - Carries the signal to the hardware
- Guardian – Instinct
  - When a module crashes, it acts as emergency break

Sequence Diagrams

- Depicts the use case of the autonomous vehicle detecting a traffic light



Limitations and Lessons Learned

Limitation: several days after we had delegated the tasks required for this assignment, one member of the group dropped the course

Lesson Learned: how to rearrange tasks and priorities on short notice

Limitation: autonomous vehicles are a relatively new form of technology, and it was not one that most of us were familiar with

Lesson Learned: the ways in which this architecture differs from what we would expect in a familiar piece of software

Example: from a security NFR perspective, the implementation of software for detecting malicious instructions for the vehicle exhibits an interesting dynamic

Conclusion

- Summarizing the conceptual architecture of Apollo, the system is designed to facilitate the advancements within the autonomous driving sector where it provides the infrastructure needed accomplish this momentous endeavour
- The system is built on a framework that allows for high power component concurrency that is also dynamic in nature. Developers can add new modules and expect low latency and high throughput for the data communication
- There is a big emphasis on security, not only from the standpoint of maintaining the integrity of the data and limiting its exposure, but to also preserving the safety of other participants on the road
- The architecture is built on the Process-Control style, where the different components produce information that the controller system uses to adjust outputs to the vehicle

References

- Apollo Open Platform. Apollo. (n.d.). Retrieved February 14, 2022, from https://apollo.auto/developer.html
- Apollo perception. Apollo. (n.d.). Retrieved February 14, 2022, from https://apollo.auto/platform/perception.html
- Behere, S., &amp; Törngren, M. (2015, December 29). A functional reference architecture for autonomous driving. Information and Software Technology. Retrieved February 14, 2022, from https://www.sciencedirect.com/science/article/abs/pii/S0950584915002177?via%3Dihub
- Kumar, N. (2020, December 27). 7 data challenges in autonomous driving. Medium. Retrieved February 14, 2022, from https://medium.datadriveninvestor.com/7-data-challenges-in-autonomous-driving-e21d05dacc3a
- Apollo Cyber RT framework. Apollo. (n.d.). Retrieved February 14, 2022, from https://apollo.auto/cyber.html
- Apollo Cyber Security. Apollo. (n.d.). Retrieved February 14, 2022, from https://apollo.auto/platform/security.html
- ApolloAuto/apollo: An open autonomous driving platform. GitHub. (n.d.). Retrieved February 13, 2022, from https://github.com/ApolloAuto/apollo
- Apollo/modules/canbus at master · Apolloauto/apollo. GitHub. (n.d.). Retrieved February 13, 2022, from https://github.com/ApolloAuto/apollo/tree/master/modules/canbus
- Apollo/modules/control at master · Apolloauto/apollo. GitHub. (n.d.). Retrieved February 13, 2022, from https://github.com/ApolloAuto/apollo/tree/master/modules/control
- Apollo/modules/dreamview at master · Apolloauto/apollo. GitHub. (n.d.). Retrieved February 13, 2022, from https://github.com/ApolloAuto/apollo/tree/master/modules/dreamview
- Apollo/modules/localization at master · ApolloAuto/apollo. GitHub. (n.d.). Retrieved February 13, 2022, from https://github.com/ApolloAuto/apollo/tree/master/modules/localization
- Apollo/modules/monitor at master · Apolloauto/apollo. GitHub. (n.d.). Retrieved February 13, 2022, from https://github.com/ApolloAuto/apollo/tree/master/modules/monitor
- Apollo/modules/perception at master · Apolloauto/apollo. GitHub. (n.d.). Retrieved February 13, 2022, from https://github.com/ApolloAuto/apollo/tree/master/modules/perception
- Apollo/modules/planning at master · Apolloauto/apollo. GitHub. (n.d.). Retrieved February 13, 2022, from https://github.com/ApolloAuto/apollo/tree/master/modules/planning
- Apollo/modules/prediction at master · Apolloauto/apollo. GitHub. (n.d.). Retrieved February 13, 2022, from https://github.com/ApolloAuto/apollo/tree/master/modules/prediction