Group 8 - Artemis

•••

Conceptual Architecture of Apollo

Video Link: https://www.youtube.com/watch?v=CTVOG51Uo6Y

Team Artemis

- Josh Otten - Conceptual Architecture, Subsystems, Use cases
- Aleks Jugovic - Presenter, Subsystems, Use cases, Subsystems, Derivation Process
- Muyun Yang - Proofreading, Referencing
- Chong Guan - Introduction, Abstract, Conclusion
- Daniel Jang - Presenter, Subsystems, Conceptual Architecture, Lessons Learned
- Wooseok Lee - Group Leader, Lessons Learned, Conceptual Architecture, Subsystems

Agenda

Introduction

Apollo

- Open source, high performance, flexible architecture for autonomous vehicles
- Road perception
- Obstacle detection
- Collision and accident avoidance

Derivation Process

Derivation Process

- Three stepped approach:

1. Individual Brainstorming

2. Consolidation of individual ideas

3. Finalisation of subsystems, interactions, and style

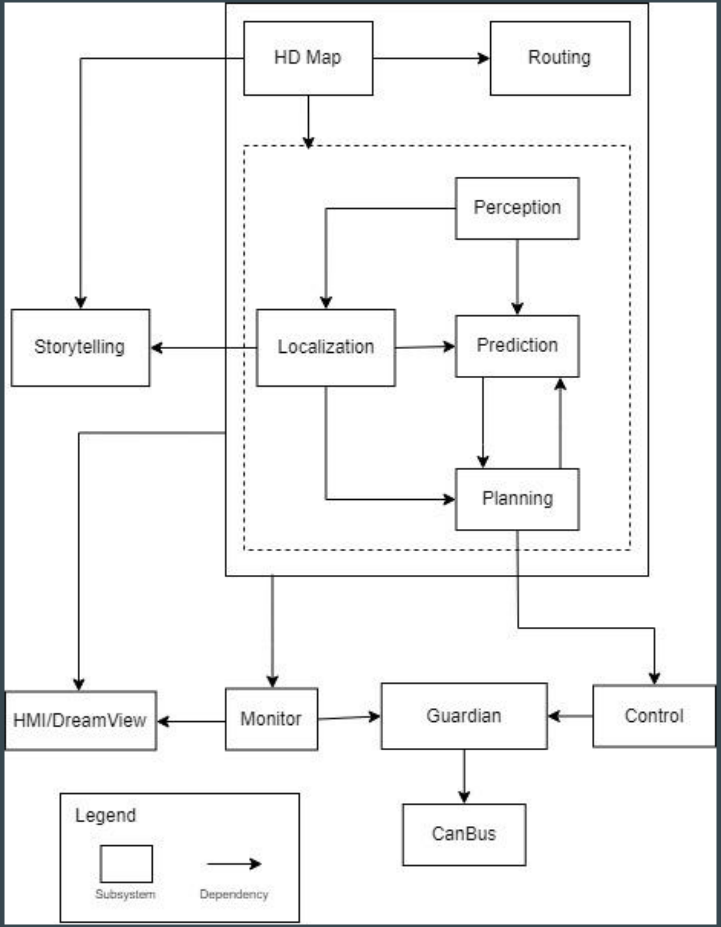
Top Level Architecture

Conceptual Architecture

Publish and Subscribe

Flexible

Low Concurrency



Subsystems

Perception

- Identifies the world around the car
  - Detects, classifies, tracks, and predicts motion of obstacles

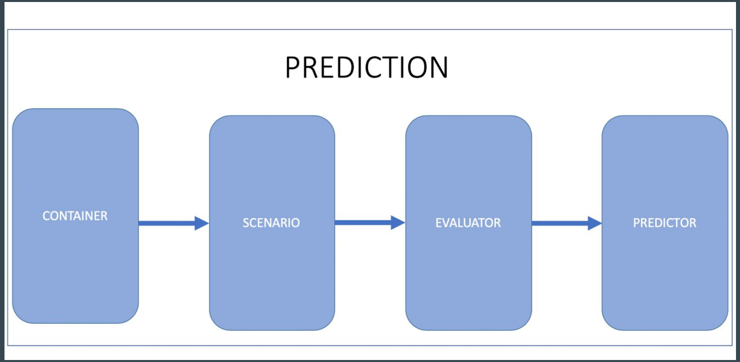
- Inputs:
  - Camera, LidAR, radar data

- Outputs:
  - 3D labelled obstacle data

Prediction

- Predicts the behaviour of obstacles identified by the Perception module

- Inputs:
  - 3D obstacle data from the Perception module



- Outputs:
  - Predicted movement of obstacles to be used by the Planning module

Routing

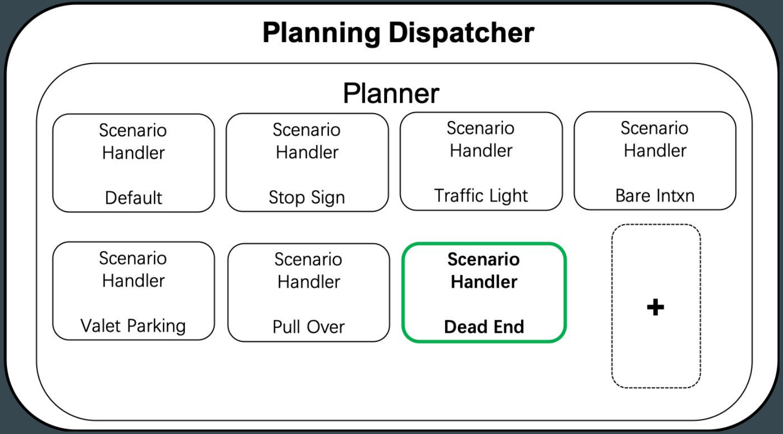
- Generates the high-level navigation based on a routing topology.

- Inputs:
  - Map data
  - Vehicle start and end location

- Outputs:
  - Routing data

Planning

- Finds the trajectory for the vehicle to take
- Inputs:
  - Localization data, object detection from Perception, object predicted behaviour from Prediction
- Outputs:
  - Vehicle trajectory data


Planning Dispatcher
Planner
Scenario Handler Default
Scenario Handler Stop Sign
Scenario Handler Traffic Light
Scenario Handler Bare Intxn
Scenario Handler Valet Parking
Scenario Handler Pull Over
Scenario Handler Dead End
+

Control

- Executes the planned trajectory via commands sent to the CanBus
  - Throttle, brake, steering
- Inputs:
  - Planned trajectory from Planning, localization data from Localization
- Outputs:
  - Control commands for the chassis of the car

CanBus

- Sends the control commands to the hardware of the vehicle using the CanBus standard

- Inputs:
  - Control commands from Control, and guardian commands from Guardian

- Outputs:
  - Chassis information back to Control for feedback

HD Map

- Map data library queried by modules

- Outputs:
    - Map data to Localization, Perception, Prediction, and Planning modules

Localization

- Estimates the current location of the vehicle

- Inputs:
  - Various sensor information from Perception module
  - Map data from HD Map

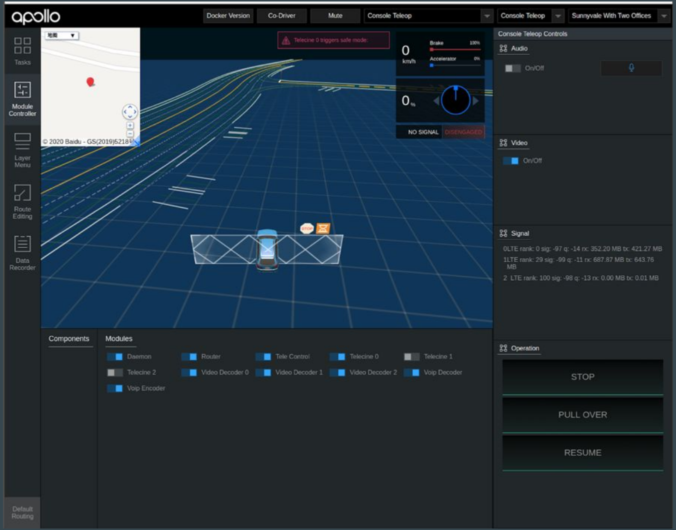
- Outputs:
  - Location of the vehicle to be used by Prediction and Planning modules

HMI (Dreamview)

- Web application to allow driver to visualise car data and interact with individual modules
  - Turn them on and off, view their status, and toggle self driving

- Inputs:
  - Control commands from Control
  - Guardian commands from Guardian

- Outputs:
  - Chassis information back to Control for feedback



Telecine 0 triggers safe mode:

0LTE rank: 0 sig: -97 q: -14 rx: 352.20 MB tx: 421.27 MB
1LTE rank: 29 sig: -99 q: -11 rx: 687.87 MB tx: 643.76 MB
2 LTE rank: 100 sig: -98 q: -13 rx: 0.00 MB tx: 0.01 MB

Monitor

- Surveillance system for Apollo’s software modules and hardware components
- Detects failures and notifies Guardian

- Inputs:
  - Data from all modules

- Outputs:
  - System information to HMI
  - Failure detection to Guardian

Guardian

- Safety module for the vehicle.
- Upon error, interrupts control instructions and brings car to a safe stop

- Inputs
  - Failure notification from Monitor
  - Control instructions from Control

- Outputs
  - Emergency stop instructions to CanBus

Storytelling

- High-level, global scenario manager which manages potential scenarios the vehicle can be in

- Inputs:
  - Localization and map data

- Outputs:
  - Scenarios

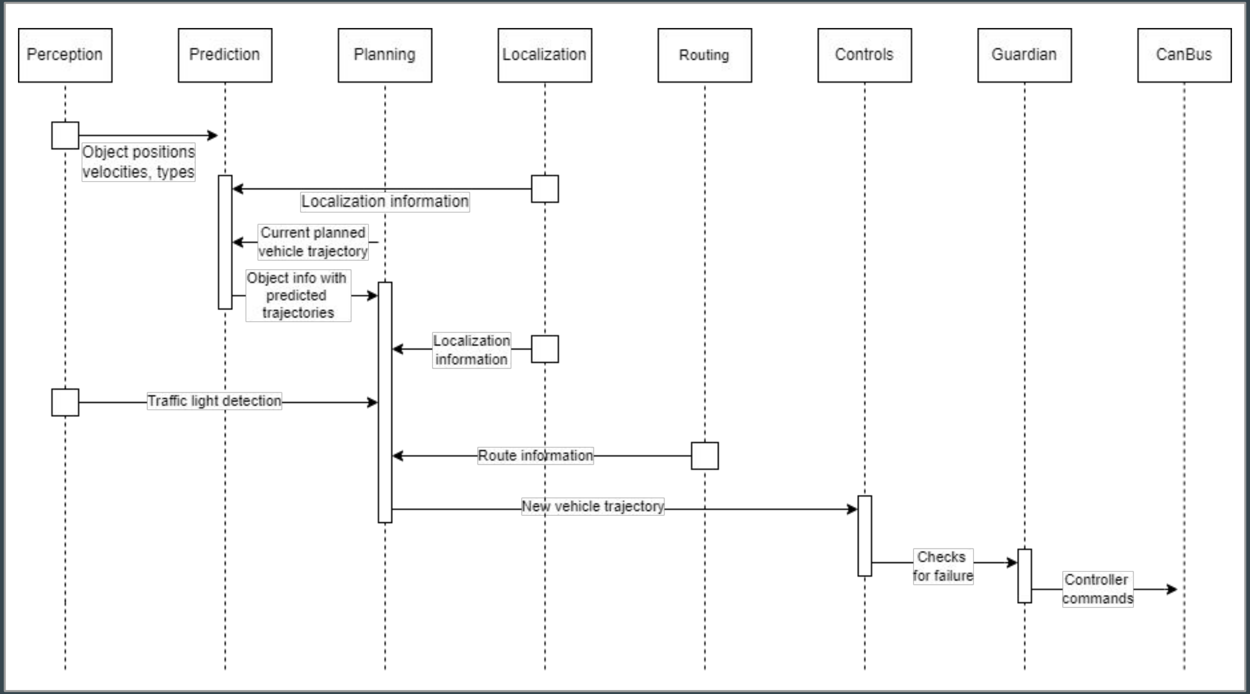
Use Cases

Use Case: Normal Driving

Most common use

No abnormal conditions

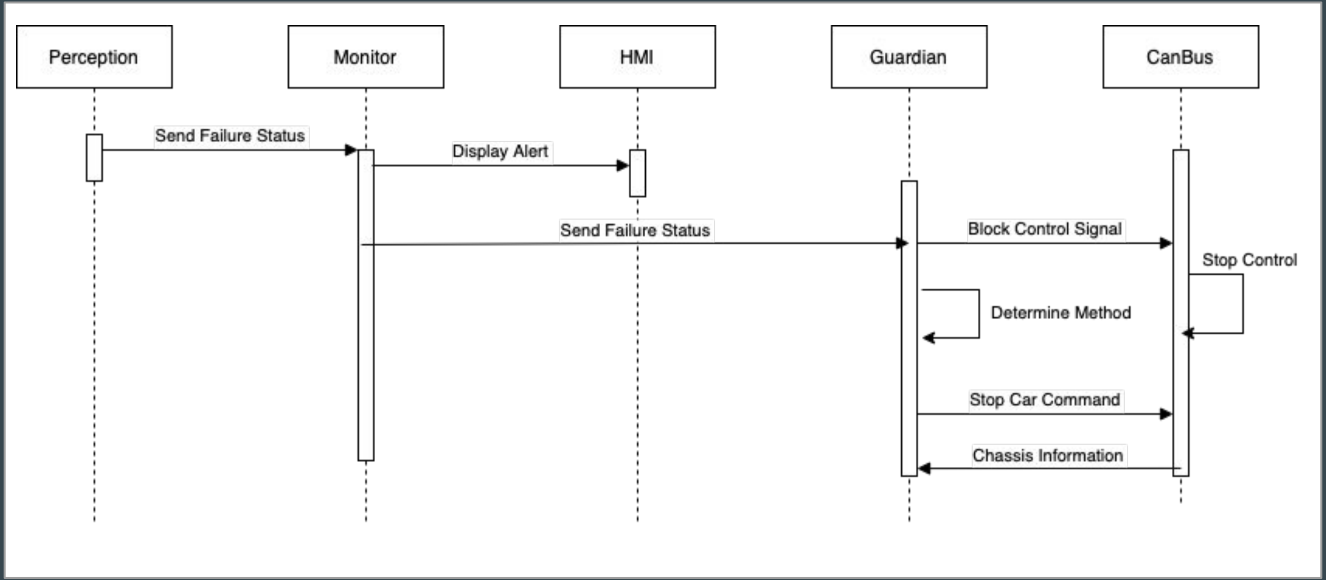
No present dangers



Use Case: Emergency Stop

Module crash or hardware failure

Exhibits the Guardian module functionality



Lessons Learned

Limitation and Lessons Learned

- Reference between release version documentation
- Clearly understand what is required to complete
- Always have organized and up-to-date documentation
- Overcoming gaps in documentation

Conclusion

Conclusion

- Open source, high performance, flexible architecture for autonomous vehicles
- Publish and Subscribe architecture
- Subsystems
- For the future