OpenPilot System Architecture Analysis

by: Team Horizon

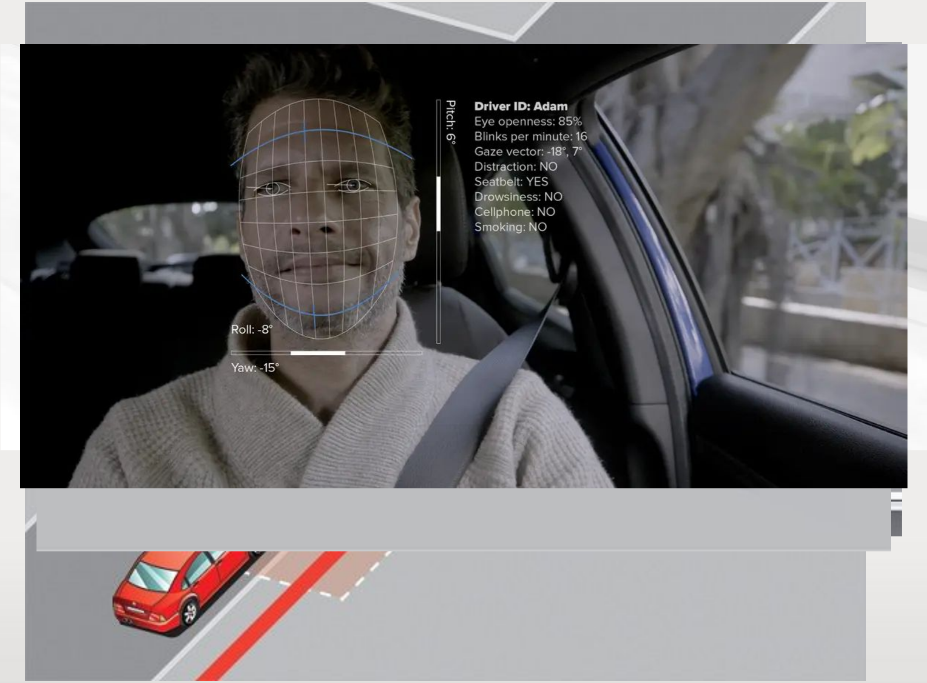
YORK U

Agenda

1. OpenPilot Overview
2. Features of OpenPilot
3. Components of OpenPilot
4. Architecture Styles
5. Concurrency
6. System Evolution
7. Modelling Alternatives
8. Trade Offs

Functionality of the top level systems

Overview & Features of OpenPilot



- Adaptive Cruise Control (ACC)
- Automated Lane Centering (ALC)
- Forward Collision Warning (FCW)
- Lane Departure Warning (LDW)
- Driver Monitoring (DM)

Identifying Subsystems

Subsystems

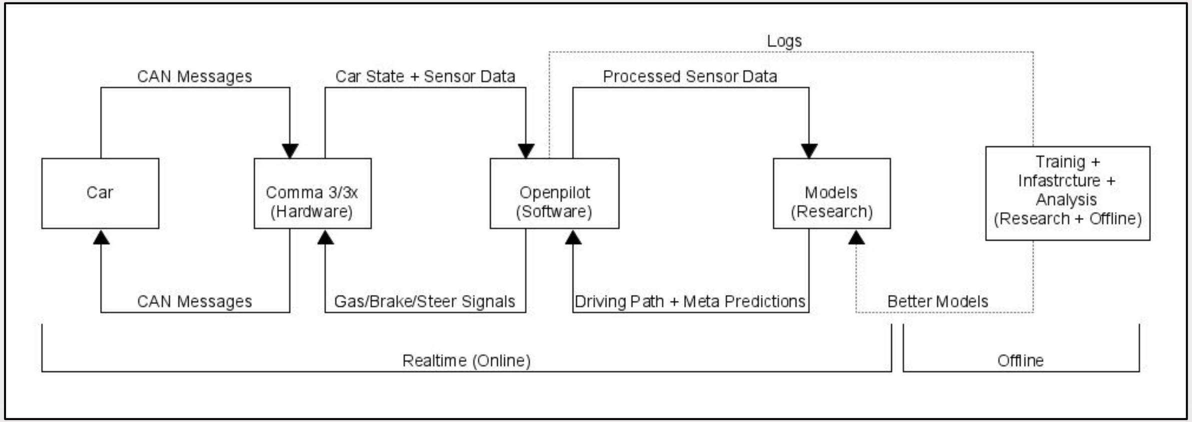
- Sensors
- Actuators
- Neural Network Routers
- Localization
- Calibration
- Controls
- Logging
- Misc. services
- Hardware (panda/comma)

Architecture Styles

Expected Architecture Styles Based on Documentation Research

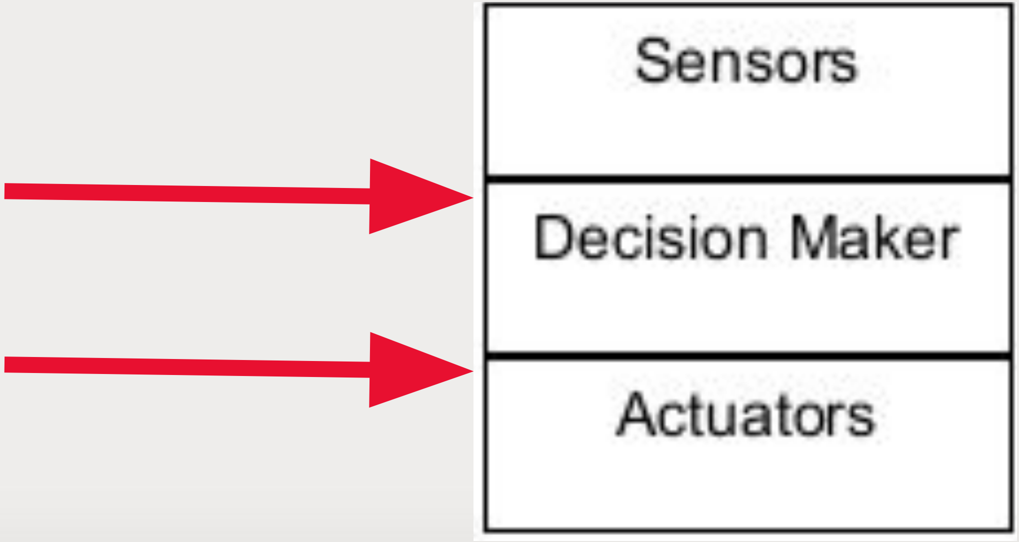
Layered

- Layers don’t communicate with non-adjacent layers.
- Each layer has its own domain.
- Lowest level being vehicle hardware (OBD-II), highest being Application.



Layered - Communication

- CAN buses
- Panda
- opendbc
- boardd
- Cereal



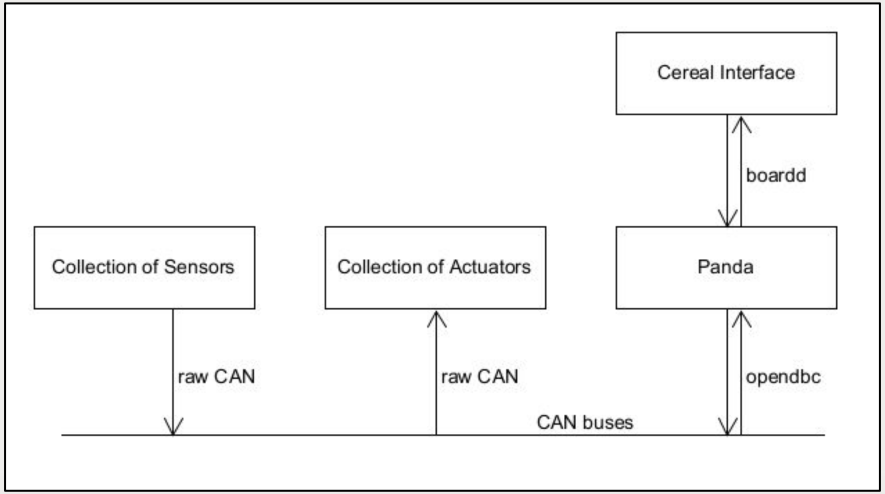
Sensors

Decision Maker

Actuators

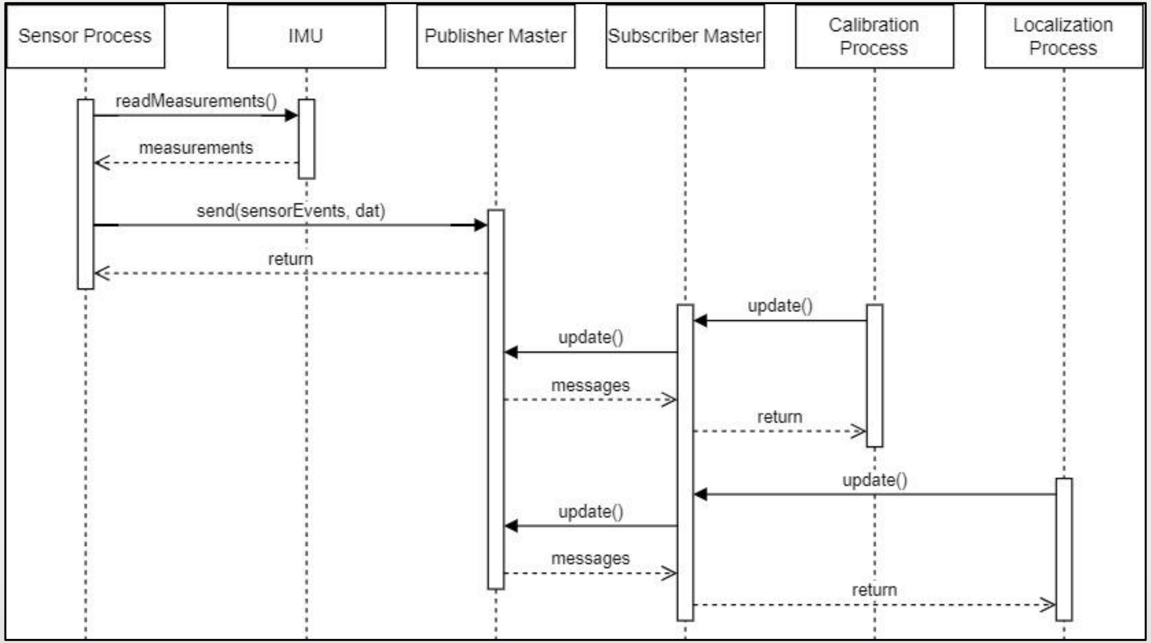
Implicit Invocation - Event Based

- Main form of communication
  - Sensors
  - Actuators
- Loosely coupled components
- Event busses



Implicit Invocation - Publish & Subscribe

- Cereal
  - Robotics systems message exchange specification
  - Interprocess communication library
- Libraries
  - ZeroMQ
  - msgq



Process Control - Closed Loop Feedback

- Maintains a specific parameter at a desired value.
- Closed Loop: Uses current value to determine which adjustments are required to meet desired value.

OpenPilot

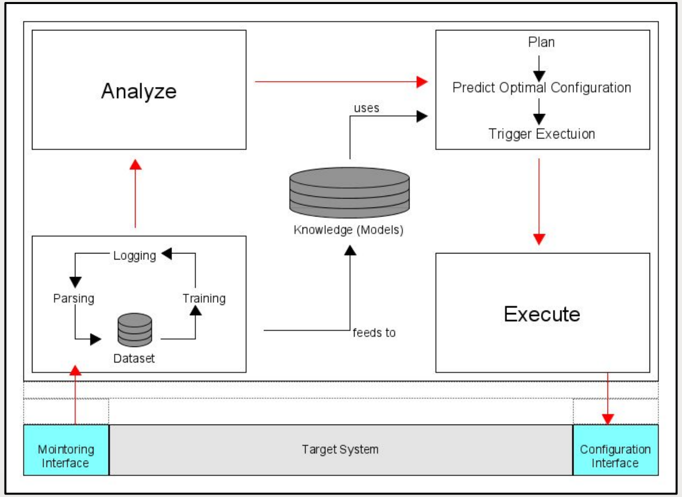
- **Adaptive Cruise Control (ACC)**: Maintains desired speed. Read current speed and make adjustments via accelerator/brake.
- **Lane Keeping Assistant System (LKAS):** Maintains vehicle in desired position. Reads current position using cameras/sensor, and makes adjustments via steering.

Process Control - MAPE-K

- **Monitor-Analyze-Plan-Execute** over shared **Knowledge**
- Variation that uses a ML Model as Knowledge

OpenPilot

- ***Laneless Mode:*** Uses ML to predict 'where humans would normally drive'.
- Not informed about traffic laws, intersections,lanes, etc.
- Goal is to create smoother, more comfortable driving, as well as allow for unpredictable scenarios



Prediction as to how they incorporate ML, without viewing Source Code.

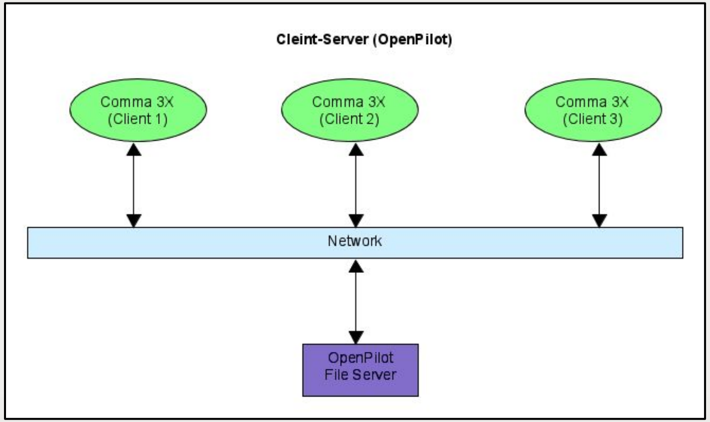
[***Image Source***]

Client-Server

- Standalone remote component (server). Server accessed by clients that make call via network.

OpenPilot

- By default, all driving data is uploaded to their servers, and used to train ML models to improve OpenPilot.
- Server accessed by Comma 3X (client) via LTE or WiFi



Other Styles

- Pipe and Filter
    - Very likely to have internal pipelines to handle data, as many diagrams include ordered sequence of events. (Sensor input becomes actuator output).
    - However, need to ensure they are stateless filters, unaware of up/downstream, etc. (via Source Code).
- Repository
    - Must be database somewhere when data is uploaded to server and used for training models.
    - Did not find any documentation about specific Repository.
    - Will likely discover more for Assignment 2.

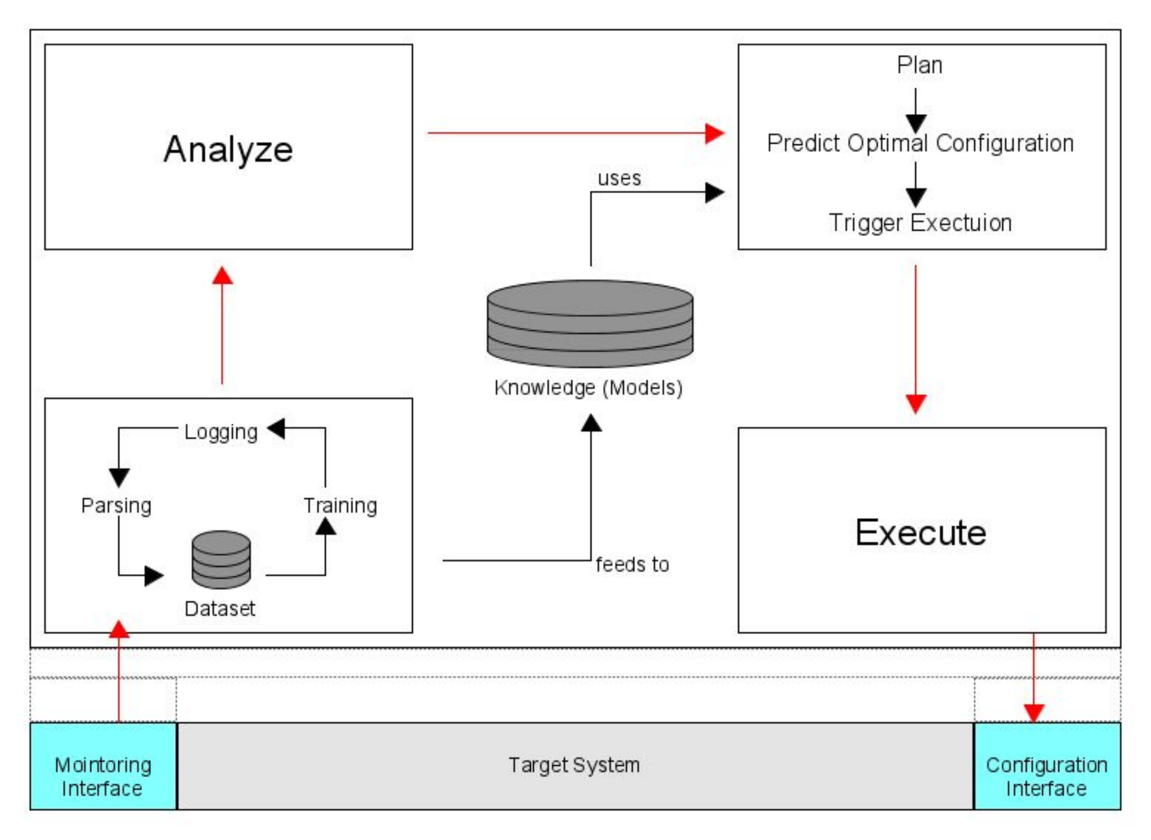
Concurrency

Concurrency

- Simultaneous input data from cameras, radar, and vehicle sensors
- Parallel decision making processing from different sensors
- Overlapping sensor input for diverse range of functionality
- Theoretical Example: Automated Lane Centering and Lane Departure Warning
    - Share similar input data however use separate decision making components for separate purposes

Concurrent processes occurring

Decision making model demonstrates that there are sub processes happening within the decision making process to deliver lane features.

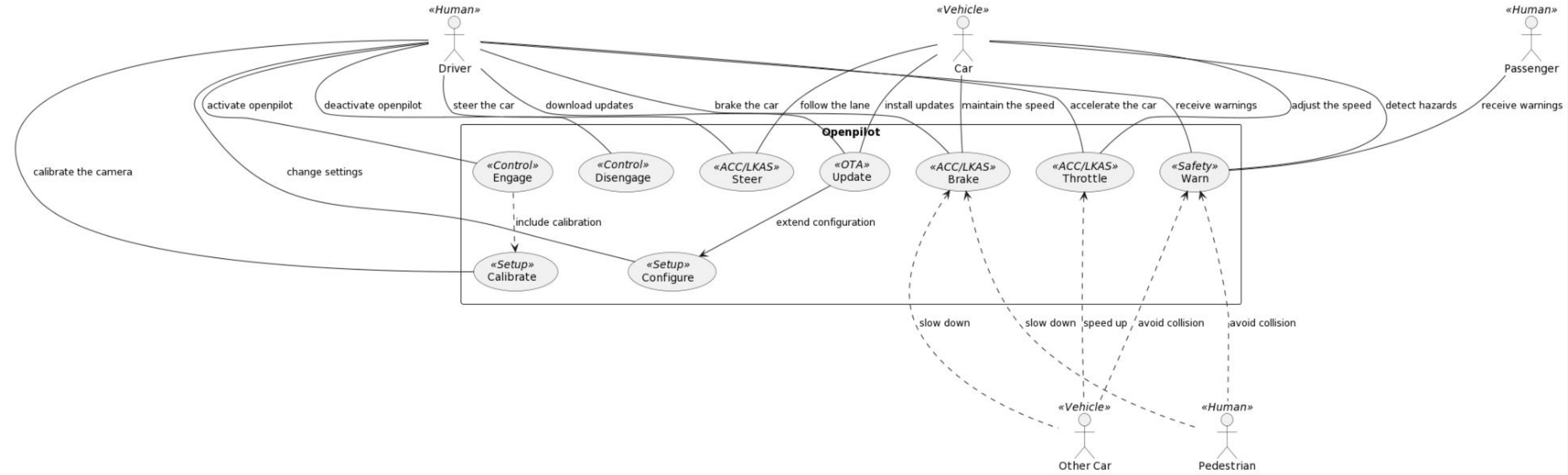


Key findings and discussion

System Evolution

- Structural release process
  - Multiple iterations enhancing and adding features
  - Example: 0.9.5 release
- Open Source Nature
  - Allows any developer to view, modify, or enhance the project
  - Enables continuous improvements through community contributions
- Contribution of internal employees
  - Diverse team such as such as Full Stack Developer, Car Interface Engineer, Production, etc
- User Involvement
  - Uploading specific data during usage
  - Leveraged by Openpilot team to improve and train better models

Use Cases


«Human»
Driver
«Vehicle»
Car
«Human»
Passenger
activate openpilot
deactivate openpilot
steer the car
download updates
brake the car
follow the lane
install updates
maintain the speed
accelerate the car
receive warnings
adjust the speed
detect hazards
receive warnings
Openpilot
calibrate the camera
change settings
«Control»
Engage
«Control»
Disengage
«ACC/LKAS»
Steer
«OTA»
Update
«ACC/LKAS»
Brake
«ACC/LKAS»
Throttle
«Safety»
Warn
include calibration
extend configuration
«Setup»
Calibrate
«Setup»
Configure
slow down
slow down
speed up
avoid collision
avoid collision
«Vehicle»
Other Car
«Human»
Pedestrian

Lessons Learned

- Compatible device needed (comma 2/3/3x)
- Not fully self-drive and limits
  - Cannot check if lane change is safe - driver must do this.
  - Requires driver awareness - otherwise alerts, or eventually will slow to a stop.
- Weather conditions
  - Not verified to function as expected in low-light, rain, fog, bright oncoming headlights, weather.
- Legality
- Telemetry and privacy
- Updates may introduce new bugs
- Interfere with manufacturers systems

Thanks for your attention!

Any questions or suggestions?