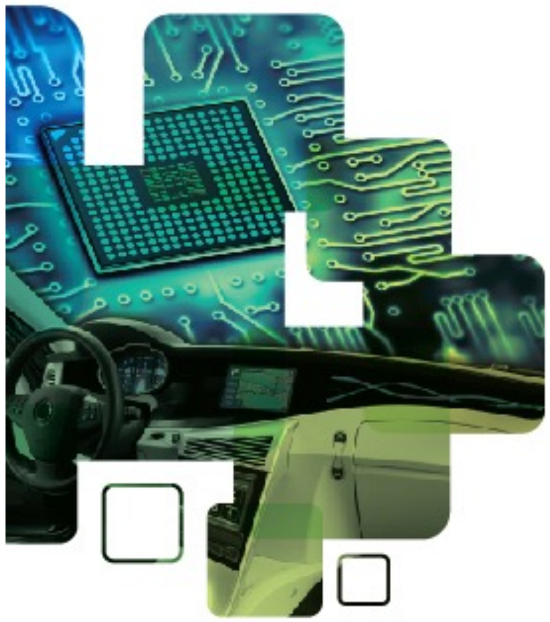


GENIVI®

AMM OPEN DAYS

Android Auto PoC

October 21

Maxim Ovchinnikov
Architect
Harman

Purpose of this PoC

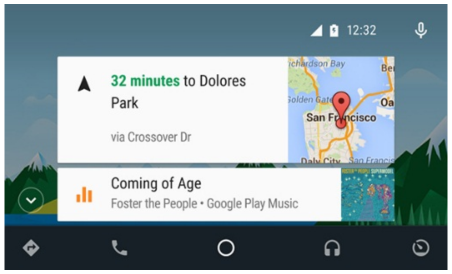
The purpose of this project was to integrate the Android Auto Projection (aka AAP or AA) stack into the GENIVI development platform (GDP).

Why Projection?

- We live in connected world and people want to stay connected while they are driving
- One of the solutions is a projection from a connected device
- Projection is secure.
- It is ready to be used immediately. No setup, no profiles creations.

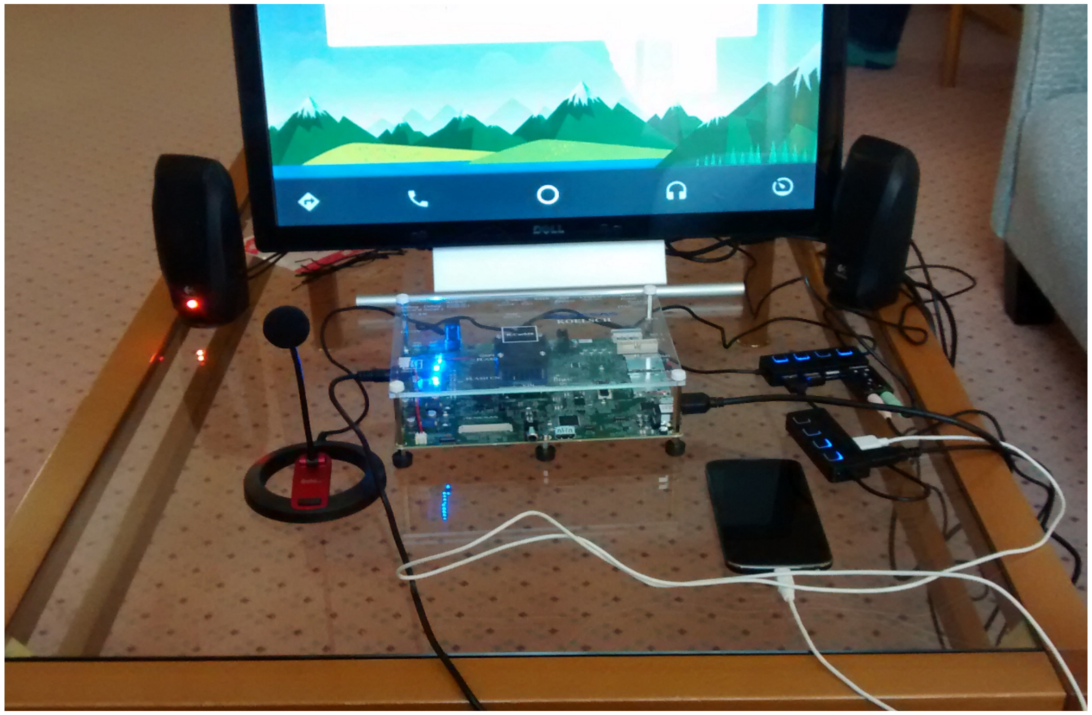
Android Auto

Android Auto is an app that integrates with your car to make it easier for you to use some of the main features of your Android phone while driving. You can control things like navigation, music, and your phone's dialer from your car's digital display so you can stay focused on the road.

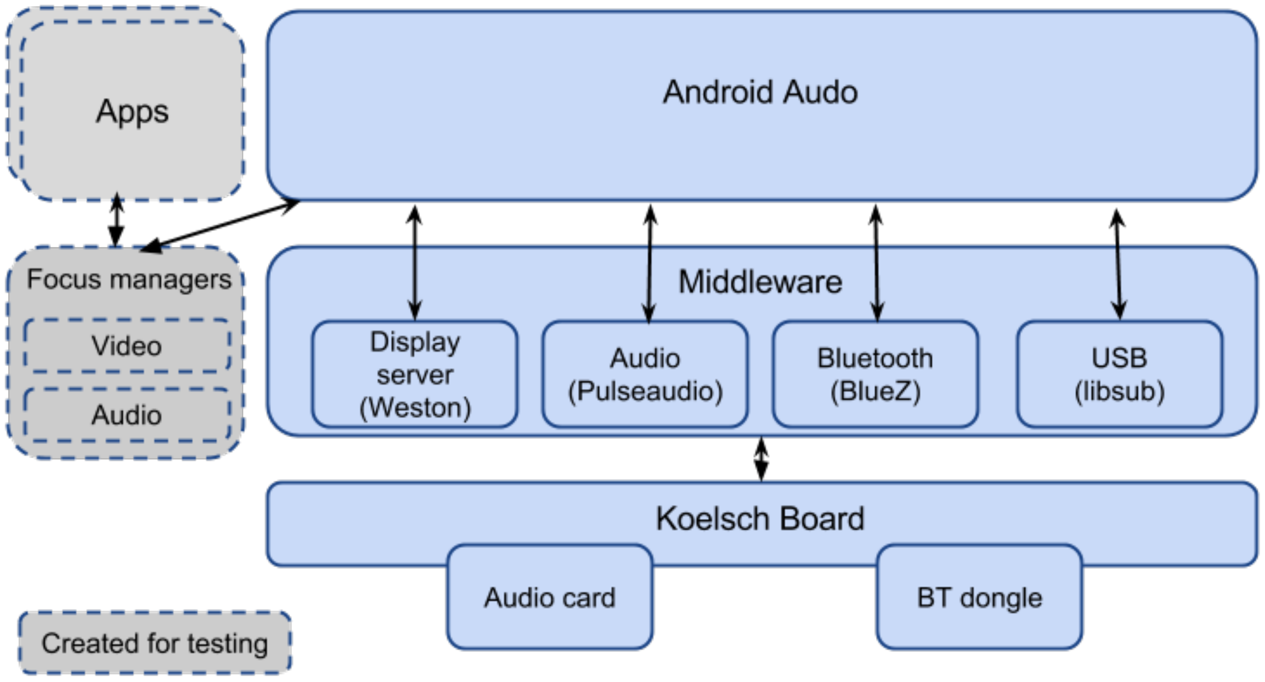

12:32
32 minutes to Dolores Park
via Crossover Dr
San Francisco
Coming of Age
Foster the People • Google Play Music

Use of Android Auto requires licensing agreement with Google. Please work with your Google Technical Area Manager (TAM) to secure appropriate licensing agreement.

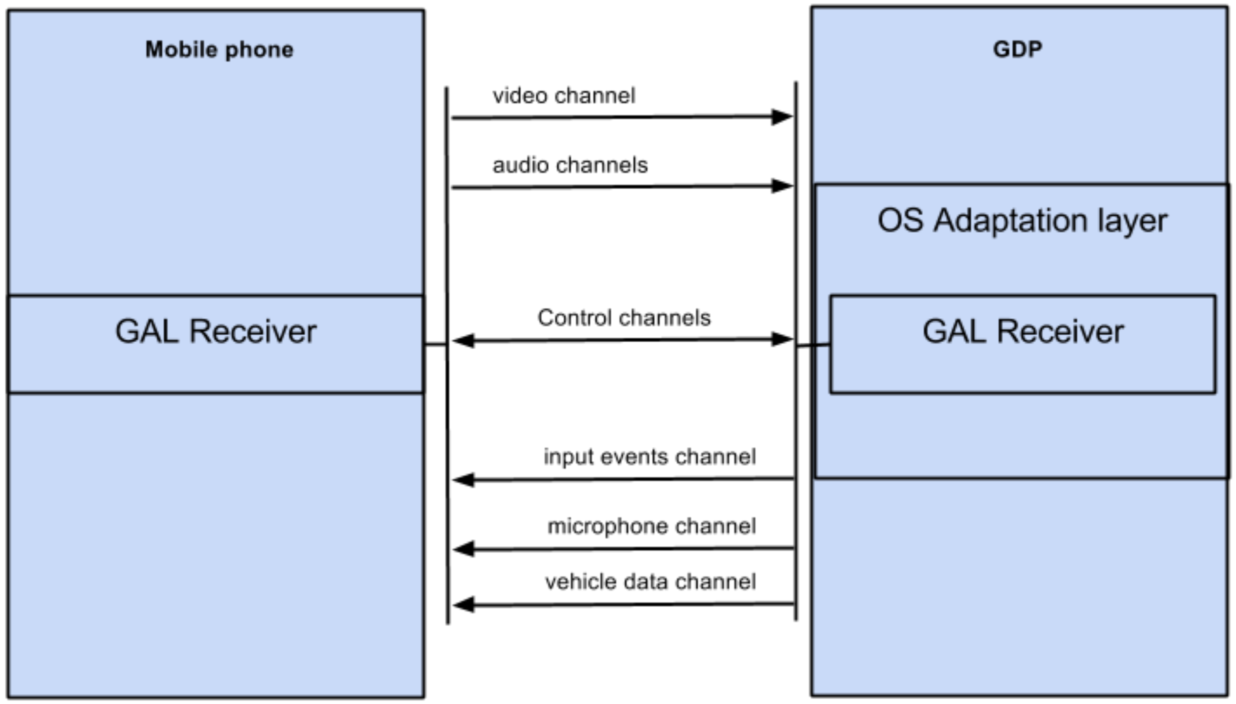
Hardware setup



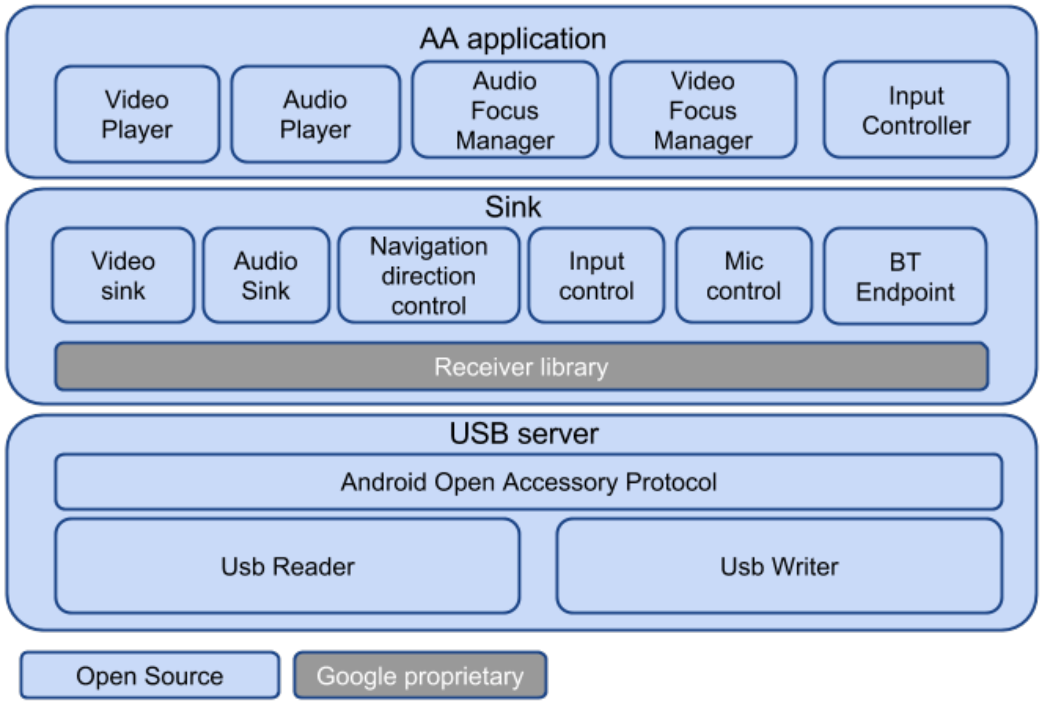
System breakdown



Data flow


Mobile phone
GAL Receiver
video channel
audio channels
Control channels
input events channel
microphone channel
vehicle data channel
GDP
OS Adaptation layer
GAL Receiver

Internal architecture



USB server – connects to AA phone over AOAP protocol (on top of USB), exchanges the raw data with Sink layer

Sink – acts as a proxy server between GDP and AA phone. Google proprietary Receiver Library simplifies communication between sink and sources.

AA application – outputs the data (video, audio, navigation etc.) to the user and sends input data (touch, keys, voice) down the stack to the phone. Application is a QT based

GENIVI

Implementation: usb-server

- Responsible for initializing USB connection
- Listens to libusb events for detecting phone connect
- USB reading & writing
- Starts up the sink if it an Android Auto phone
- Main point of interaction during startup for the system

Implementation: sink

- Responsible for starting up audio sinks for:
  - Navigation
  - Music
  - System audio
- Starts up audio source for microphone
- Starts up video sink, which displays video data from the phone (the phone's “display”)
- Starts up input sources:
  - Touchscreen input (rotary controller, d-pad etc.)
- Starts up Bluetooth endpoint
- Communicates with the usb-server using pipes

Implementation: media-server

- Qt/QML app, theoretically enabling addition of UI elements/overlays
- Qt enables easy-to-use touch input
- Used for both audio and video
    - h.264 for video (Gstreamer pipeline)
    - AAC for audio (Gstreamer pipeline), or PCM for audio (Raw PulseAudio input)
- Integrates with Audio and Video Focus Manager
- Communicates with the sink using a named pipe

Implementation: media-server-mic

- Started for each recording session
- Records PCM data from PulseAudio
- Communicates with sink using pipes

Implementation: Audio Focus Mgr.

- Was created for testing but this should be a system service
- Responsible for keeping track of the app currently holding audio focus
- Informs apps on loss/gain of audio focus
- Supports different kinds of audio focus, such as:
  - Transient focus (focus will be held for a short amount of time)
  - Media focus (potentially held for a longer amount of time)
  - Focus which may be ducked (other apps may cause us to lower our volume)
- Each application has a different priority level and type, used for prioritization
- Could potentially be used as a controller plug-in for Audio Manager

Implementation: Video Focus Mgr.

- Was created for testing but this should be a system service
- Keeps track of the app currently holding video focus
- Interfaced over D-Bus
- Only has two states; focused or not focused
- Should be replaced/integrated in to UI framework

Implementation: BlueZ pairing agent

- Was created for testing but this should be a system service
- Launched upon starting Bluetooth communication
- Called by BlueZ to verify Android Auto phone Bluetooth connection
- Fulfills the BlueZ interface for pairing agents

Implementation: Licensing

- Components developed during this project:
  - Open source, most likely under the MPL v.2

- Low-level protocol components developed by Google
  - Closed source, only available to Open Automotive Alliance members

- Instructions for combining the two will be available, and “dropping” the closed component in to the open source code will be made as easy as possible

Implementation: Next steps

- Proper integration in to the GDP
  - Currently, we use a new layer (meta-genivi-aa-demo), extending meta-genivi-demo
  - We use a custom image (meta-genivi-aa-platform), which extends meta-genivi-demo
  - Most (all?) things already compatible with the GDP, as it was used as a base
  - Integration is mostly a question of maintaining uniformity within the GDP
- Discussions will follow with GDP maintainers
- Possibly, the layer will be converted in to a package group

Testing

- Performed first step in AA licensing process – Projected Compatibility Test Suite (PCTS)
- Created 3 simple test applications (Launcher, Phone and Media) and Audio and Video Focus Managers to pass the most part of PCTS testing.
- PCTS navigation and sensors tests were skipped.

Code publishing

- All code except Google Receiver Library will be available to download after Google will review and approve it. It was approved once but due to delay with signing the ELA this must be done again ☹

- Open Automotive Alliance Members who have access to Receiver Library can download it from Google repository, integrate into PoC in order to build it.

Design Challenges

- Voice call doesn’t work – no telephony audio, SCO protocol is not working correctly
- No applications for PCTS testing in GDP 7. No audio focus manager is in the platform.
- Microphone works with external audio card only

Questions?