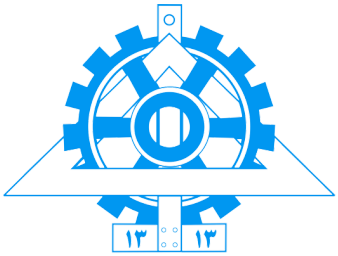


فاز دوم پروژه نهایی

دانشور امراللهی (۸۱۰۱۹۷۶۸۵)
علیرضا آقایی (۸۱۰۱۹۷۶۷۹)
مهیار کریمی (۸۱۰۱۹۷۶۹۰)
سینا کمالی (۸۱۰۱۹۷۵۶۹)

بهار ۱۴۰۱

فهرست مطالب

مقدمه **3**

نصب ابزارها **3**

دریافت فایل‌های مورد نیاز از مخزن‌ها GitHub 3

نصب packageهای مورد نیاز 3

نصب aasdk 4

نصب openauto 7

اجرای سناریوی پیش‌فرض **9**

۱. حالت USB 10

۲. حالت Wireless 11

حالت :WiFi 11

حالت Bluetooth 12

مقدمه

در این فاز ما به نصب openauto و پخش یک موزیک از طریق USB و Wifi با استفاده از آن می‌پردازیم. این فاز بر خلاف سادگی ظاهری‌ای که ممکن است داشته باشد، مشکلات بسیاری را برای ما ایجاد کرد که در ادامه آن‌ها را بررسی می‌کنیم و راه حل‌های آنها را هم ذکر می‌کنیم.

نصب ابزارها

در این بخش ما به نصب کتابخانه و ابزارهای مورد نیاز برای اجرای openauto می‌پردازیم. در هر بخش به صورت کامل مشکلات مواجه شده و راه حل های آنها آورده شده است.
در این بخش دستورات مورد نیاز برای build را از
https://github.com/mynetz/openauto/wiki/Build-instructions---Ubuntu-18.04
دنبال خواهیم کرد.

دریافت فایل‌های مورد نیاز از مخزن‌ها GitHub

برای build کردن aasdk، ابتدا فایل‌ها را از آدرس https://github.com/opencardev/aasdk دریافت می‌کنیم.
در همان directory ای که aasdk را دریافت کرده‌ایم، به صورت موازی، فایل‌های openauto را نیز از آدرس https://github.com/opencardev/openauto دریافت می‌کنیم. این عملیات با دستور git clone قابل انجام است.

نصب packageهای مورد نیاز

به طور کلی، وابستگی‌های نرم‌افزاری ابزارهای aasdk, openauto به دسته‌های زیر قابل تقسیم است:

- کامپایل پرونده‌های protocol buffer
- بخش‌هایی از کتابخانه‌ی boost[1]
- بسته‌های نرم‌افزاری مانند librtaudio-dev برای ارتباط با سخت‌افزار
- کتابخانه‌ها و محیط اجرای Qt

```
sudo apt install g++ cmake protobuf-compiler libprotobuf-dev \
    libboost-system-dev libboost-log-dev libboost-test-dev \
    libusb-1.0-0-dev libssl-dev librtaudio-dev libjack-jackd2-dev \
    qtbase5-dev qtmultimedia5-dev qtconnectivity5-dev
```

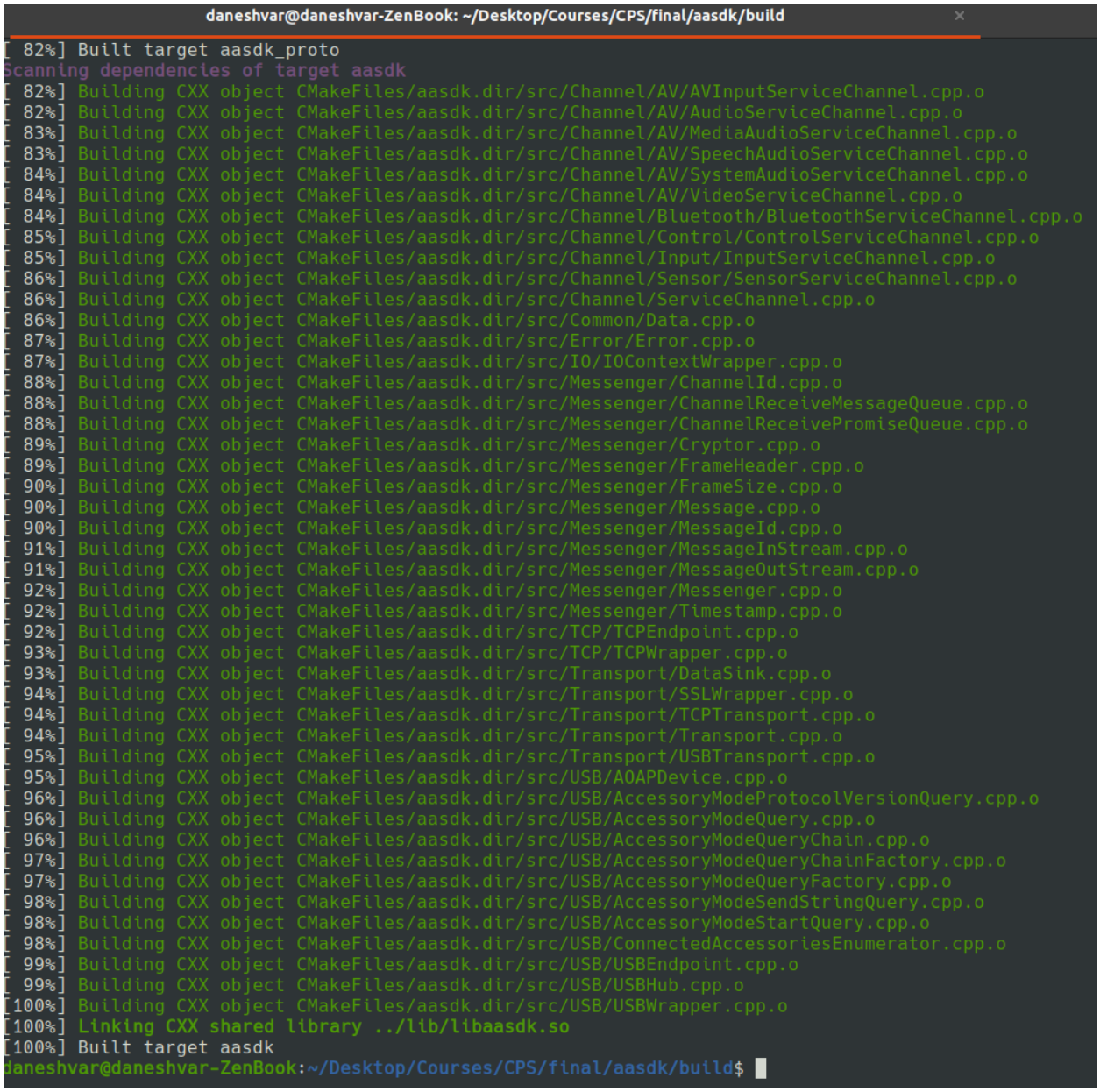
[1] این کتابخانه امکانات افزوده‌ای برای توسعه‌ی نرم‌افزار به کمک زبان C++ را فراهم می‌کند.

نصب aasdk

ابتدا branch را به development تغییر می‌دهیم. نهایتا به ترتیب دستور cmake و make وارد می‌کنیم.

```
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/aasdk/build$ git
branch
* newdev
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/aasdk/build$ git
checkout development
Branch 'development' set up to track remote branch 'development' from
'origin'.
Switched to a new branch 'development'
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/aasdk/build$ git
branch
* development
  newdev
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/aasdk/build$ cmake
-DCMAKE_BUILD_TYPE=Release ..
-- The CXX compiler identification is GNU 9.4.0
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Protobuf: /usr/lib/x86_64-linux-gnu/libprotobuf.so;-lpthread
(found version "3.6.1")
-- Found Boost:
/usr/lib/x86_64-linux-gnu/cmake/Boost-1.71.0/BoostConfig.cmake (found
version "1.71.0") found components: system log unit_test_framework
-- Found libusb-1.0:
--   - Includes: /usr/include/libusb-1.0
--   - Libraries: /usr/lib/x86_64-linux-gnu/libusb-1.0.so
-- Found OpenSSL: /usr/lib/x86_64-linux-gnu/libcrypto.so (found version
"1.1.1f")
-- Configuring done
-- Generating done
-- Build files have been written to:
/home/daneshvar/Desktop/Courses/CPS/final/aasdk/build
```

```
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/aasdk/build$ make
[  1%] Running cpp protocol buffer compiler on
/home/daneshvar/Desktop/Courses/CPS/final/aasdk/aasdk_proto/WifiSecurityRes
ponseMessage.proto
.
.
.
[100%] Building CXX object CMakeFiles/aasdk.dir/src/USB/USBWrapper.cpp.o
[100%] Linking CXX shared library ../lib/libaasdk.so
[100%] Built target aasdk
```



شکل(۱): بخشی از مراحل build کردن aasdk

خروجی‌های کامل دستور make در فایلی به اسم aasdk-built.txt در کنار گزارش پروژه آپلود شده است.

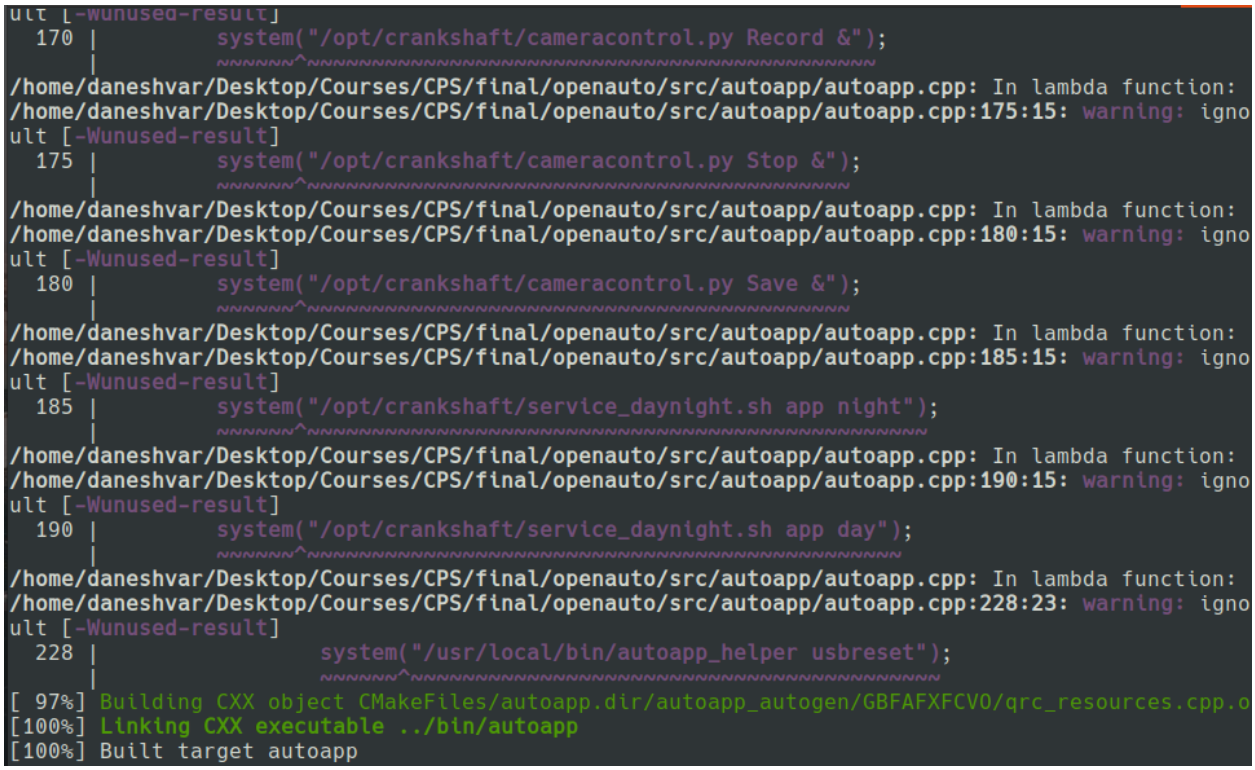
نصب openauto

برای غیرفعال‌سازی GPS، با استفاده از دستور git checkout به یکی از commitهای قبلی پروژه بازمی‌گردیم که GPS را support نمی‌کرد. سایر مراحل مشابه بخش مربوط به aasdk است و بایستی دستورهای cmake, make را اجرا کنیم.

```
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/openauto$ git
branch
* crankshaft-ng
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/openauto$ git
checkout fd3a00a8e3ecb430838ad263be883eed57d95a37
Note: switching to 'fd3a00a8e3ecb430838ad263be883eed57d95a37'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to
false

HEAD is now at fd3a00a [QT] QT 5.15 fixes for alignment errors
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/openauto$ ls
assets  CMakeLists.txt  cmake_modules  include  Readme.md  src
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/openauto$ mkdir
build
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/openauto$ cd build/
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/openauto/build$
cmake -DCMAKE_BUILD_TYPE=Release
-DAASDK_INCLUDE_DIRS="$PWD/../../aasdk/include"
-DAASDK_LIBRARIES="$PWD/../../aasdk/lib/libaasdk.so"
-DAASDK_PROTO_INCLUDE_DIRS="$PWD/../../aasdk/build"
```

```
-DAASDK_PROTO_LIBRARIES="$PWD/../../aasdk/lib/libaasdk_proto.so" ..
-- The CXX compiler identification is GNU 9.4.0
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Boost:
/usr/lib/x86_64-linux-gnu/cmake/Boost-1.71.0/BoostConfig.cmake (found
version "1.71.0") found components: system log unit_test_framework
-- Found libusb-1.0:
--   - Includes: /usr/include/libusb-1.0
--   - Libraries: /usr/lib/x86_64-linux-gnu/libusb-1.0.so
-- Found Protobuf: /usr/lib/x86_64-linux-gnu/libprotobuf.so;-lpthread
(found version "3.6.1")
-- Found OpenSSL: /usr/lib/x86_64-linux-gnu/libcrypto.so (found version
"1.1.1f")
-- Found rtaudio:
--   - Includes: /usr/include/rtaudio
--   - Libraries: /usr/lib/x86_64-linux-gnu/librtaudio.so
-- Found taglib:
--   - Includes: /usr/include/taglib
--   - Libraries: /usr/lib/x86_64-linux-gnu/libtag.so;-ltag
-- Found blkid:
--   - Includes: /usr/include/blkid
--   - Libraries: /usr/lib/x86_64-linux-gnu/libblkid.so
-- Configuring done
-- Generating done
-- Build files have been written to:
/home/daneshvar/Desktop/Courses/CPS/final/openauto/build
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/openauto/build$
make
Scanning dependencies of target btservice_autogen
[  2%] Automatic MOC and UIC for target btservice
[  2%] Built target btservice_autogen
Scanning dependencies of target btservice
[  4%] Building CXX object
CMakeFiles/btservice.dir/btservice_autogen/mocs_compilation.cpp.o
.
.
.
[ 97%] Building CXX object
```

```
CMakeFiles/autoapp.dir/autoapp_autogen/GBFAFXFCVO/qrc_resources.cpp.o
[100%] Linking CXX executable ../bin/autoapp
[100%] Built target autoapp
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/openauto/build$
```



شکل(۲): بخشی از مراحل build کردن openauto

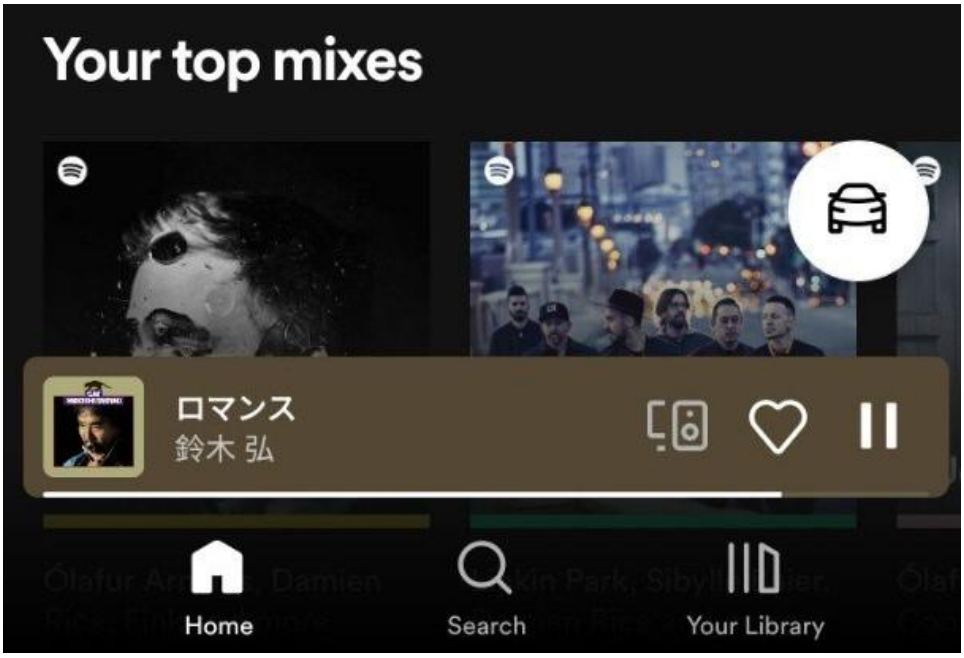
اجرای سناریوی پیش‌فرض

در این مرحله، پس از ساختن aasdk و openauto تلاش می‌کنیم که تلفن همراه را با دو روش USB و بی‌سیم[2] به openauto متصل کنیم. در هر دو سناریوی زیر فرض می‌کنیم که ابزار openauto در حال اجراست. برای اجرای openauto بایستی autoapp را از پوشه openauto/bin اجرا کنیم.

[2] wireless

۱. حالت USB

پس از این‌که تلفن همراه را به کمک USB به کامپیوتر متصل می‌کنیم، ابزار Android Auto روی تلفن همراه به‌طور خودکار اجرا می‌شود:

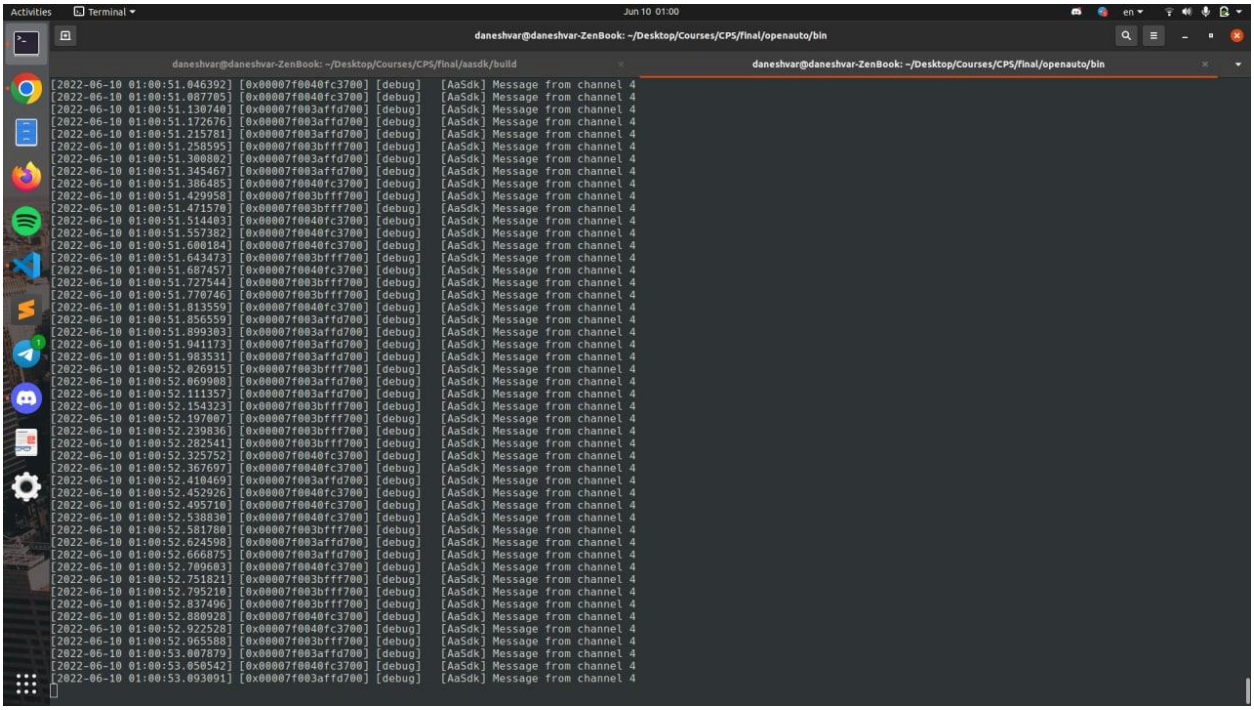


شکل(۳): همانطور که در سمت راست شکل مشخص است، spotify متوجه وصل شدن ما به یک ماشین شده است

پس از پایان پیکربندی[3] Android Auto روی تلفن، در صورت اجرای پرونده‌ای صوتی روی تلفن، سیگنال صوتی از طریق مکانیزم پیام‌رسانی[4] به کامپیوتر منتقل می‌شود و در نهایت، خروجی صوتی از طریق کامپیوتر قابل شنیدن است.

[3] configuration

[4] Message passing



شکل(۴): تصویری از لاگ open auto در هنگام پخش موزیک. دقت کنید که channel 4 همان کانال پخش audio میباشد.

۲. حالت Wireless

در این بخش با استفاده از Bluetooth و WiFi پخش آهنگ صورت می‌گیرد.

حالت WiFi:

آی‌پی گوشی از بخش status information گوشی قابل مشاهده است. بعد از اجرای دستور tcpip اتصال گوشی با کابل usb به کامپیوتر بایستی قطع شود.

```
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/openauto/bin$ adb devices
List of devices attached
RFCR8102WQM     device

daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/openauto/bin$ adb tcpip 5555
restarting in TCP mode port: 5555
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/openauto/bin$ adb connect 192.168.1.53:5555
connected to 192.168.1.53:5555
daneshvar@daneshvar-ZenBook:~/Desktop/Courses/CPS/final/openauto/bin$
```

شکل(۵): اتصال با WiFi به گوشی

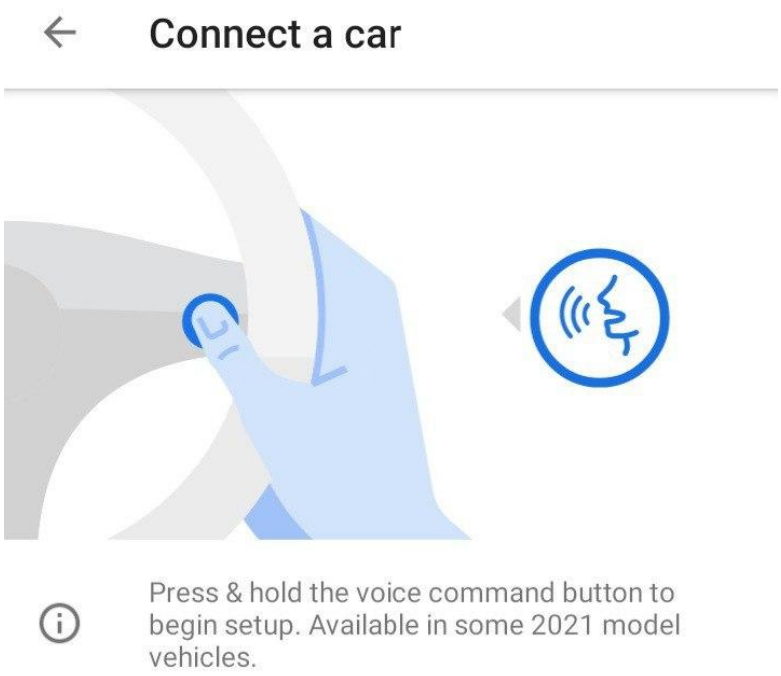
این مرحله کاملا مشابه فاز ۱ است.

حالت Bluetooth

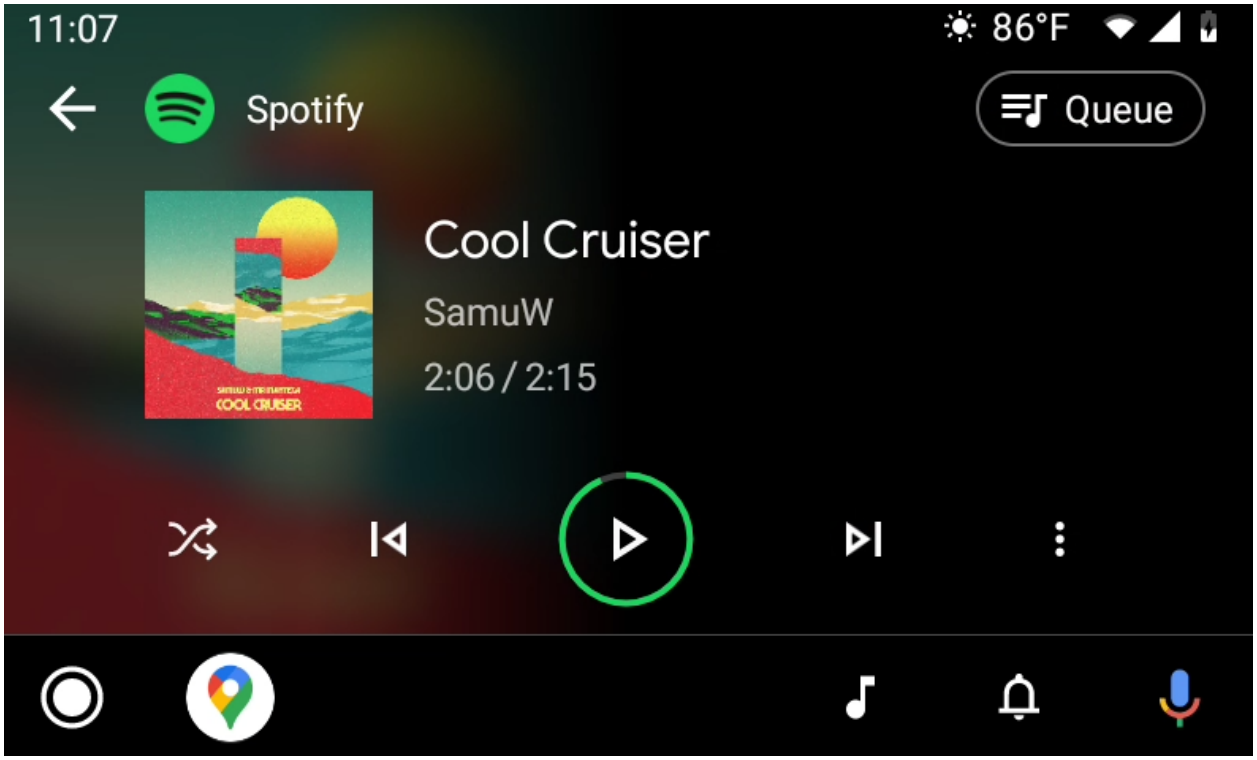
در فاز قبلی پروژه، مراحل اتصال تلفن و کامپیوتر از طریق WiFi و ابزار adb را توضیح داده‌ایم. در این فاز، برای اتصال بی‌سیم بین تلفن و کامپیوتر از اتصال بلوتوث استفاده می‌کنیم که توسط خود ابزار Android Auto روی تلفن پشتیبانی می‌شود.

پس از ایجاد اتصال بلوتوث بین تلفن و کامپیوتر، صفحهٔ اصلی openauto به صورت زیر خواهد بود:

شکل(۶): تصویر اجرای فایل اجرایی autoapp. همانطور که در بالا راست تصویر مشخص است، اتصال از طریق bluetooth صورت گرفته است.



شکل(۷): تنظیمات بلوتوث در نرم‌افزار android auto در گوشی



شکل(۸): نمونه پخش آهنگ