1 DMS Cross-compilation for OKT507

Building a C++ application (`dms_app`) on an Ubuntu x86_64 PC, but the **output binary must run on OKT507**, which uses an **ARM64 (aarch64) processor**.

This is called **cross-compilation**.

Key components involved:

Component	Purpose
Ubuntu Host PC	Build machine (x86_64)
OKT507 Board	Target machine (ARM64)
Cross Compiler	Builds ARM binaries on x86
Sysroot	Target filesystem (libs, headers)
OpenCV (ARM)	Computer vision library for OKT507
CMake	Build system generator
Make	Builds the final binary

---

2 Directory Layout (Actual Setup)

```
/home/vishwali_vijay/
│
├── dms_haar/                      ← DMS application source
│   ├── main.cpp
│   ├── audio.cpp
│   ├── CMakeLists.txt
│   └── build/                     ← Build directory
│
├── opencv-okt507-install/         ← ARM OpenCV install
│   ├── include/opencv4/
│   ├── lib/
│   └── lib/cmake/opencv4/
│
├── okt507-sysroot/                ← Target filesystem
│   ├── lib/
│   ├── usr/lib/
│   └── usr/include/
│
└── opencv-okt507-toolchain.cmake ← Cross-compile config
```

---

3 Why Sysroot Is Mandatory

The **sysroot** represents the **OKT507 root filesystem**.

It contains:

- `libc`, `libpthread`
- System headers
- Runtime libraries

Without sysroot:
Linker errors
GLIBC mismatches
Binary won’t run on board

You correctly used:

```
SET(CMAKE_SYSROOT /home/vishwali_vijay/okt507-sysroot)
```

---

4 Cross Compiler Used

```
/usr/bin/aarch64-linux-gnu-gcc
/usr/bin/aarch64-linux-gnu-g++
```

These produce **ARM64 binaries**, not x86.

Verify:

```
aarch64-linux-gnu-g++ --version
```

---

5 Toolchain File (MOST IMPORTANT FILE)

`opencv-okt507-toolchain.cmake`

This tells CMake:

- Target OS
- Target CPU
- Which compiler to use
- Where the sysroot is

Final Correct Toolchain File

```
SET(CMAKE_SYSTEM_NAME Linux)
SET(CMAKE_SYSTEM_PROCESSOR aarch64)

SET(CMAKE_C_COMPILER   /usr/bin/aarch64-linux-gnu-gcc)
SET(CMAKE_CXX_COMPILER /usr/bin/aarch64-linux-gnu-g++)

SET(CMAKE_SYSROOT /home/vishwali_vijay/okt507-sysroot)

SET(CMAKE_FIND_ROOT_PATH /home/vishwali_vijay/okt507-sysroot)

SET(CMAKE_FIND_ROOT_PATH_MODE_PROGRAM NEVER)
SET(CMAKE_FIND_ROOT_PATH_MODE_LIBRARY ONLY)
SET(CMAKE_FIND_ROOT_PATH_MODE_INCLUDE ONLY)
SET(CMAKE_FIND_ROOT_PATH_MODE_PACKAGE ONLY)

SET(CMAKE_C_FLAGS   "--sysroot=/home/vishwali_vijay/okt507-sysroot")
SET(CMAKE_CXX_FLAGS "--sysroot=/home/vishwali_vijay/okt507-sysroot")
```

---

6 OpenCV Cross-Compilation

- Cross-compiled OpenCV for ARM
- Installed it into: `~/opencv-okt507-install`

Key verification:

```
find ~/opencv-okt507-install -name OpenCVConfig.cmake

Found at:lib/cmake/opencv4/OpenCVConfig.cmake
```

---

7 CMakeLists.txt

`dms_haar/CMakeLists.txt`

```
cmake_minimum_required(VERSION 3.10)
project(dms_app)

set(CMAKE_CXX_STANDARD 14)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

find_package(OpenCV REQUIRED)

add_executable(dms_app
    main.cpp
    audio.cpp
)

target_include_directories(dms_app PRIVATE
    ${OpenCV_INCLUDE_DIRS}
)

target_link_libraries(dms_app
    ${OpenCV_LIBS}
    pthread
)
```

Important points:

- No hardcoded paths
- No manual OpenCV include paths
- Uses `OpenCVConfig.cmake`
- Links pthread explicitly (fixes linker errors)

---

8 Full Build Flow (FINAL)

Step 1: Clean build

```
cd ~/dms_haar/build
rm -rf *
```

Step 2: Configure

```
cmake .. -DCMAKE_TOOLCHAIN_FILE=~/opencv-okt507-toolchain.cmake
```

Output you can see:

```
-- Found OpenCV: /home/.../opencv-okt507-install
-- OpenCV include dirs: .../include/opencv4
-- OpenCV libraries: opencv_core;opencv_imgproc;...
```

Step 3: Build

```
make -j$(nproc)
```

Final success:

```
[100%] Built target dms_app
```

---

9 Resulting Binary

```
file dms_app
```

Output:

```
ELF 64-bit LSB executable, ARM aarch64
```

This confirms **correct cross-compilation**
