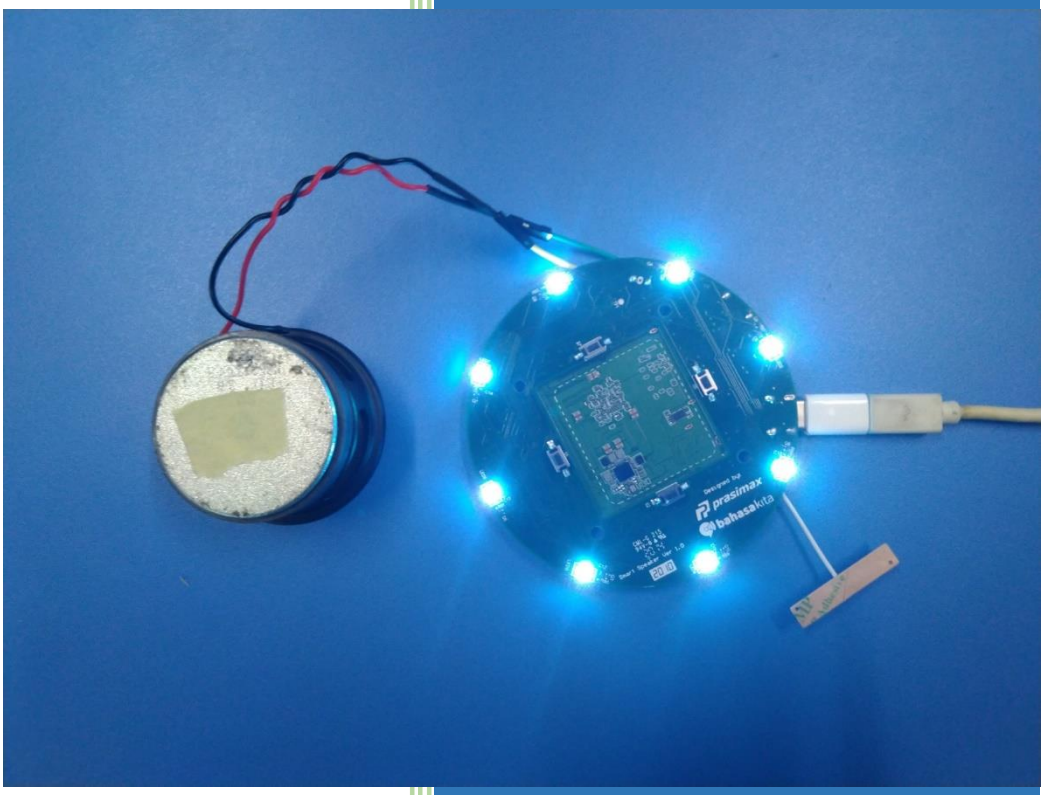
2021

Panduan Software Smart Speaker



Isro Bayu F.

PRASIMAX

01/01/2021

Build Image QCS404

Tahap pertama yang harus dikuasai dalam mengerjakan software smart speaker adalah build image. Untuk dapat build image, ikuti panduan pada dokumen-dokumen berikut:

1. 80-yb405-129 : QCS40x Software Programmer's Guide
2. 80-yb420-4 : QCS40x SSRD System Bring Up Guide

Dokumen-dokumen tersebut dapat diperoleh dari Create Point Qualcomm. Panduan lengkap dapat ditemukan pada dokumen nomor 1, dimana setiap langkah dijelaskan secara terperinci untuk setiap versi terbaru yang dirilis. Panduan pada dokumen nomor 2 sifatnya lebih sederhana tetapi hanya spesifik untuk versi LE.1.0 dan LE.1.1 sedangkan versi terbaru adalah LE.1.2 untuk versi stabil dan LE.1.3 untuk versi dalam pengembangan.

Untuk mempermudah proses build image, source code versi LE.1.2 yang diperlukan telah dikumpulkan dan dapat diunduh dari repository berikut:

```
https://gitlab.com/didipmx/smart-speaker/-/tree/isro
```

Source code tersebut telah melalui proses build aplikasi non-HLOS dan hanya perlu melalui proses build aplikasi HLOS supaya dapat digunakan dalam build image. Perlu diperhatikan bahwa aplikasi non-HLOS menargetkan prosesor Qualcomm QCS404 dengan ROM NAND flash 512MB, RAM LPDDR2 512MB, dan modul Wifi + BT WCN3980. Apabila terdapat perbedaan, aplikasi non-HLOS perlu melalui proses build ulang.

Ketika build image sudah berhasil dilakukan, image dapat dipasang pada board EVK Qualcomm untuk pengujian fungsionalitas. Apabila board EVK Qualcomm tidak tersedia, maka pengujian dapat dilakukan pada board custom yang menggunakan prosesor Qualcomm QCS404. Untuk dapat melakukan pemasangan image pada board custom, diperlukan beberapa penyesuaian seperti mengubah konfigurasi driver.

Ada beberapa level dalam mengubah konfigurasi driver. Level yang paling dasar adalah dengan cara menerapkan driver yang telah disediakan oleh Qualcomm pada file device tree. Hal ini dapat dilakukan apabila komponen yang digunakan adalah komponen yang telah direkomendasikan oleh Qualcomm. Panduan untuk mengubah file device tree dan menerapkan perubahan driver yang telah disediakan Qualcomm dapat ditemukan pada dokumen berikut:

1. 80-yb420-13 : QCS400 Linux Audio Interface
2. 80-yb420-14 : QCS400 Linux Device Driver

Level yang lebih lanjut adalah dengan membuat driver sendiri dari sebuah komponen kemudian menerapkannya pada file device tree. Untuk bisa membuat driver sendiri, tentunya butuh expertise dalam memahami cara kerja komponen dan standar pembuatan atau struktur fungsi driver supaya dapat berjalan pada sistem operasi. Beberapa vendor mungkin sudah menyediakan driver dari komponen yang mereka buat. Jika demikian, masih diperlukan expertise dalam memahami struktur fungsi driver sebelum driver tersebut dapat diterapkan pada device tree.

Prasimax telah berhasil menyelesaikan board custom DVT1. Board tersebut menggunakan komponen-komponen yang telah direkomendasikan Qualcomm dengan desain minimum sehingga tidak diperlukan banyak perubahan pada device tree. Panduan untuk mengubah isi file device tree untuk board custom DVT1 Prasimax dapat ditemukan pada dokumen berikut:

SSD01_Device-Tree.pdf : Modifikasi Device Tree QCS404

Dokumen tersebut dapat diperoleh dari repository Prasimax.

Menambahkan Aplikasi ke dalam Image QCS404

Ketika image sudah teruji secara fungsionalitas, pengguna dapat menambahkan aplikasi ke dalam image. Untuk menambahkan aplikasi ke dalam image, pengguna harus membuat file bitbake recipe, menambahkan file bitbake recipe ke layer yocto, dan melakukan build ulang. Ada 2 jenis aplikasi yang dapat ditambahkan melalui file bitbake recipe, yaitu:

1. Aplikasi dalam bentuk file source code.
   File source code biasanya dibuat dengan menggunakan bahasa pemrograman C/C++ dan memanfaatkan library open source yang dapat diperoleh dari internet. File source code ini nantinya akan melalui proses build aplikasi sebelum proses build image dijalankan. Proses build aplikasi dapat dilakukan secara otomatis maupun manual, tergantung dari definisi dan deklarasi metode build yang diberikan dalam file bitbake recipe. Dalam file bitbake recipe, metode build aplikasi setidaknya terdiri dari empat tahap.
   a. Tahap pertama adalah configure
      Pada tahap ini, proses build akan mengumpulkan setiap tools yang diperlukan untuk build aplikasi seperti cmake atau make. Tools yang digunakan perlu didefinisikan secara jelas dalam file bitbake recipe. Build secara otomatis akan mengumpulkan tools sesuai konfigurasi yocto, tetapi build secara manual akan memilah tools berdasarkan deklarasi dalam file bitbake recipe. Deklarasi ini biasanya diperlukan apabila aplikasi memerlukan versi tools yang lebih lama dari versi yang digunakan dalam konfigurasi yocto ataupun tools memerlukan konfigurasi parameter tertentu sebelum dapat digunakan untuk build aplikasi.
   b. Tahap kedua adalah compile
      Pada tahap ini, proses build akan mengubah file source code menjadi file executable dengan menjalankan perintah sesuai tools yang digunakan. Build secara otomatis akan menjalankan perintah sesuai konfigurasi yocto, tetapi build secara manual akan menjalankan perintah berdasarkan deklarasi dalam file bitbake recipe. Deklarasi ini biasanya diperlukan apabila proses build memerlukan parameter tambahan seperti flag atau lain-lain dalam menjalankan perintah.
   c. Tahap ketiga adalah install
      Pada tahap ini, file library akan dikumpulkan berdasarkan dependency file executable. Build secara otomatis akan mengumpulkan library sesuai konfigurasi yocto, tetapi build secara manual akan mengumpulkan library berdasarkan deklarasi dalam file bitbake recipe. Deklarasi ini biasanya diperlukan apabila library yang diperlukan memiliki versi yang tidak sama dengan versi dalam konfigurasi yocto atau library static yang perlu disalin secara manual karena telah di-build secara terpisah.
   d. Tahap keempat adalah package
      Pada tahap ini, seluruh file yang dihasilkan dalam proses build aplikasi termasuk file executable beserta file library dependency akan diverifikasi baik versi maupun kesesuaian arsitekturnya dengan image yang hendak di-build.

   Ada kalanya deklarasi dari pengguna harus dijalankan diantara keempat metode tersebut sehingga pengguna perlu menggunakan metode prepend dan append. Untuk memahami metode build dalam file bitbake recipe, pengguna dapat mempelajari dokumen Yocto Project Development Manual yang tersedia di internet.
2. Aplikasi dalam bentuk file executable terkompresi.
   Aplikasi dalam bentuk file executable terkompresi sudah melalui proses build aplikasi di sistem lain dan menyertakan semua library static sehingga siap digunakan. Namun, file executable dan file library static ini masih perlu disalin pada tahap ketiga (install) dan masih perlu menambahkan file library dynamic jika ada. File-file ini kemudian akan diverifikasi pada tahap keempat (package) untuk memastikan bahwa versi maupun arsitekturnya sesuai dengan image yang hendak di-build. Kelebihan dari aplikasi jenis ini adalah kerahasiaan source code tetap terjaga dan pengguna dapat menggunakan bahasa pemrograman apapun yang diinginkan asalkan pengguna dapat menghasilkan file executable dan mengumpulkan seluruh library dependency.

Menambahkan Aplikasi Bahasa Kita ke dalam Image QCS404

Aplikasi Bahasa Kita adalah aplikasi dalam bentuk file source code. Semua file dibuat dengan bahasa pemrograman C. Aplikasi Bahasa Kita terdiri dari 3 aplikasi, yaitu bahasakita-voiceui, interfacing-device-ss, dan ipc-kws. File source code dari ketiga aplikasi tersebut adalah proprietary Bahasa Kita dan hanya bisa diakses dari repository Bahasa Kita sehingga source code yang digunakan dalam file bitbake recipe adalah file source code terkompresi. File bitbake recipe untuk ketiga aplikasi tersebut adalah sebagai berikut:

1. File bitbake aplikasi bahasakita-voiceui (bahasakita-voiceui_1.0.bb)

```
LICENSE = "CLOSED"
LIC_FILES_CHKSUM = "file://LICENSE;md5=86d3f3a95c324c9479bd8986968f4327"

SRC_URI = "\
           file://bahasakita-voiceui.tar \
          "

# Modify these as desired
PV = "1.0.0"
SRCREV = "f4bedeb9075a497c9fb6532e24b7ec9e27b5e511"

S = "${WORKDIR}/bahasakita-voiceui"

# NOTE: unable to map the following pkg-config dependencies: pulseaudio alsa-lib
#       (this is based on recipes that have previously been built and packaged)
DEPENDS = "curl libwebsockets sqlite3 openssl json-c util-linux libpcre libffi
pulseaudio alsa-lib"

# NOTE: if this software is not capable of being built in a separate build directory
# from the source, you should replace autotools with autotools-brokensep in the
# inherit line
inherit pkgconfig autotools

# Specify any options you want to pass to the configure script using EXTRA_OECONF:
EXTRA_OECONF = ""
```

Perlu diperhatikan bahwa bahasakita-voiceui.tar adalah file source code terkompresi. Tools yang diperlukan dalam proses build adalah pkgconfig dan autotools. Library dependency yang diperlukan adalah curl, libwebsockets, sqlite3, openssl, json-c, util-linux, libpcre, libffi, pulseaudio, dan alsa-lib. Aplikasi di-build secara otomatis.

2. File bitbake aplikasi interfacing-device-ss (interfacing-device-ss_git.bb)

```
LICENSE = "CLOSED"
LIC_FILES_CHKSUM = "file://LICENSE;md5=1ebbd3e34237af26da5dc08a4e440464"

SRC_URI = "\
           file://interface-v2.tar \
          "

# Modify these as desired
PV = "1.0.0"
SRCREV = "71a31044f1d490d050c352dced25e7cd30b01b73"

S = "${WORKDIR}/interface-v2"

DEPENDS = "json-c curl"

# NOTE: if this software is not capable of being built in a separate build directory
# from the source, you should replace autotools with autotools-brokensep in the
# inherit line
inherit pkgconfig autotools
```

```
# Specify any options you want to pass to the configure script using EXTRA_OECONF:
EXTRA_OECONF = ""

do_compile_prepend() {
    mkdir ${WORKDIR}/build/src
}
```

Perlu diperhatikan bahwa interface-v2.tar adalah file source code terkompresi. Tools yang diperlukan dalam proses build adalah pkgconfig dan autotools. Library dependency yang diperlukan adalah curl dan json-c. Aplikasi di-build secara otomatis dengan sedikit build secara manual melalui deklarasi do_compile_prepend(). Deklarasi ini berfungsi untuk membuat direktori/folder untuk source code yang tidak berhasil dibuat oleh proses build secara otomatis.

3. File bitbake aplikasi ipc-kws (ipc-kws_git.bb)

```
LICENSE = "CLOSED"
LIC_FILES_CHKSUM = "file://LICENSE;md5=1ebbd3e34237af26da5dc08a4e440464"

SRC_URI = "\
            file://ipc-prog.tar \
          "

# Modify these as desired
PV = "1.0.0"
SRCREV = "09da9b5515b83e8750d71228d4aa903584e6b2da"

S = "${WORKDIR}/ipc-prog"

# NOTE: if this software is not capable of being built in a separate build directory
# from the source, you should replace autotools with autotools-brokensep in the
# inherit line
inherit autotools

# Specify any options you want to pass to the configure script using EXTRA_OECONF:
EXTRA_OECONF = ""
```

Perlu diperhatikan bahwa ipc-prog.tar adalah file source code terkompresi. Tools yang diperlukan dalam proses build adalah autotools. Tidak ada library dependency yang diperlukan. Aplikasi di-build secara otomatis.

Library libwebsockets yang diperlukan adalah versi 4 sedangkan konfigurasi yocto menggunakan versi 2. Oleh karena itu, pengguna perlu menambahkan file bitbake recipe untuk libwebsockets versi 4 (libwebsockets_4.0.0.bb) sebagai berikut:

```
DESCRIPTION = "A lightweight pure C library built to use minimal CPU and memory
resources, and provide fast throughput in both directions as client or server"
HOMEPAGE = "https://libwebsockets.org/"
SECTION = "networking"
MAINTAINER = "Andy Green"

LICENSE = "LGPL2.1 & BSD-3-Clause"
LIC_FILES_CHKSUM = "file://LICENSE;md5=8c47b078124308a4e1354e8d59f606b7"

SRC_URI =
"https://github.com/warmcat/${PN}/archive/v${PV}.tar.gz;downloadfilename=${PN}-
${PV}.tar.gz"
SRC_URI[md5sum] = "2865f237f74f7d9309db828cc919d4d5"
SRC_URI[sha256sum] = "412128c465c5e8e97f043db2f9e3458f5544fad7d5b894c4ab45743c536a0bcb"

inherit cmake

FILES_${PN}-dev += "${libdir}/cmake"

EXTRA_OECMAKE = "-DCMAKE_INSTALL_PREFIX=/usr \
${@oe.utils.conditional("libdir", "/usr/lib64", "-DLIB_SUFFIX=64", "", d)} \
${@oe.utils.conditional("libdir", "/usr/lib32", "-DLIB_SUFFIX=32", "", d)} \
```

```
-DLWS_WITH_STATIC=OFF \
-DLWS_WITHOUT_TESTAPPS=ON \
-DLWS_WITHOUT_TEST_SERVER=ON \
-DLWS_UNIX_SOCK=ON"

PACKAGECONFIG ??= "openssl zlib libuv"

PACKAGECONFIG[libuv] = "-DLWS_WITH_LIBUV=ON \
-DLIBUV_LIBRARIES=${STAGING_DIR_HOST}/${libdir}/libuv.so,\
-DLWS_WITH_LIBUV=OFF,libuv,"
PACKAGECONFIG[zlib] = "-DLWS_WITH_ZLIB=ON -DLWS_WITHOUT_EXTENSIONS=OFF,\
-DLWS_WITH_SSL=OFF -DLWS_WITHOUT_EXTENSIONS=ON,zlib,"
PACKAGECONFIG[openssl] = "-DLWS_WITH_SSL=ON,-DLWS_WITH_SSL=OFF,openssl,"
```

Ketika mengunduh source code dari repository, pastikan bahwa nilai SRC_URI[md5sum] dan SRC_URI[sha256sum] sudah sesuai. Tempatkan seluruh file bitbake recipe tersebut pada lokasi berikut:

```
$<work_dir>/LE.UM.4.1.2/apps_proc/poky/meta-qti-adk/recipes-adk/voice-ui-framework/
```

$<work_dir> adalah path relative untuk directory source code Qualcomm. File bitbake recipe dapat juga ditempatkan pada lokasi lain yang dianggap lebih mudah diingat oleh pengguna. Perlu diperhatikan bahwa lokasi penempatan mungkin akan mempengaruhi layer yocto. Setelah menempatkan file bitbake recipe, tambahkan aplikasi ke dalam image dengan mengubah file berikut:

```
$<work_dir>/LE.UM.4.1.2/apps_proc/poky/build/conf/local.conf
```

Tambahkan baris berikut ke dalam file local.conf:

```
IMAGE_INSTALL_append += "bahasakita-voiceui ipc-kws interfacing-device-ss"
```

Perlu diperhatikan bahwa parameter yang ditambahkan pada baris tersebut merupakan nama file bitbake recipe yang telah ditempatkan sebelumnya. Adapun libwebsockets tidak ditambahkan karena sejak awal library tersebut sudah ada meskipun dengan versi yang berbeda. Setiap kali mengubah isi file local.conf terutama saat menambahkan aplikasi, jangan lupa untuk build ulang HLOS dan build ulang image.

Ketika aplikasi sudah berhasil ditambahkan dan diuji fungsionalitasnya, pengguna dapat membuat file service supaya file executable dari aplikasi dapat dimulai secara otomatis. File service dapat ditambahkan melalui file bitbake recipe. Selain itu, pengguna juga dapat menambahkan berbagai macam file yang diperlukan untuk mendukung aplikasi ketika sedang berjalan melalui file bitbake recipe.

Aplikasi Bahasa Kita yang sudah berhasil di-build akan menghasilkan 3 file executable yang perlu dimulai secara otomatis, yaitu pairing-mobile-apps dan get-data-api dari aplikasi interfacing-device-ss serta bahasakita-voiceui-v01 dari aplikasi bahasakita-voiceui. Dengan demikian, diperlukan 3 file service yang terdiri dari 2 file service untuk aplikasi interfacing-device-ss dan 1 file service lainnya untuk aplikasi bahasakita-voiceui. File service biasanya diberi penamaan sesuai fungsi aplikasi. File service untuk aplikasi interfacing-device-ss adalah bk-pairingapp.service dan bk-getdataapi.service sedangkan file service untuk aplikasi bahasakita-voiceui adalah bk-voiceui.service. Informasi lebih detail mengenai file service dapat dipelajari dari internet. Selain itu, untuk mendukung aplikasi Bahasa Kita diperlukan 1 file bash script bernama bk-voiceui.sh dan 1 folder bernama bk-sound yang berisi beberapa file audio dengan format wav seperti bunyi.wav, idle.wav, terkoneksi.wav, wifi.wav, dan wrong.wav. File bash script nantinya akan dijalankan dari dalam file service sedangkan file audio akan digunakan sesuai kebutuhan aplikasi.

Isi file bk-pairingapp.service adalah sebagai berikut:

```
[Unit]
Description=Bahasa Kita Pairing Mobile App Service
BindsTo=connectivity-manager.service
After=connectivity-manager.service
Wants=bk-getdataapi.service
Before=bk-getdataapi.service

[Service]
Restart=on-failure
RestartSec=10
WorkingDirectory=/data
EnvironmentFile=/tmp/dbus-session
ExecStartPre=/bin/sh /usr/bin/bk-voiceui.sh
ExecStart=/usr/bin/pairing-mobile-apps

[Install]
WantedBy=multi-user.target
```

Isi file bk-getdataapi.service adalah sebagai berikut:

```
[Unit]
Description=Bahasa Kita Get Data API Service
Requires=bk-pairingapp.service
After=bk-pairingapp.service

[Service]
Restart=on-failure
RestartSec=10
WorkingDirectory=/data
EnvironmentFile=/tmp/dbus-session
ExecStart=/usr/bin/get-data-api

[Install]
WantedBy=bk-pairingapp.service
```

Isi file bk-voiceui.service adalah sebagai berikut:

```
[Unit]
Description=Bahasa Kita VoiceUI App Service
Requires=audio-manager.service pulseaudio.service
After=voiceUI.service
Wants=bk-wakeword.service
Before=bk-wakeword.service

[Service]
Restart=on-failure
RestartSec=10
EnvironmentFile=/tmp/dbus-session
ExecStartPre=/bin/sh /usr/bin/bk-voiceui.sh
ExecStart=/usr/bin/bahasakita-voiceui-v01 202.83.120.122 8765

[Install]
WantedBy=voiceUI.service
```

Isi file bk-voiceui.sh adalah sebagai berikut:

```
if [ ! -d "/data/tmp" ]; then
  mkdir "/data/tmp"
fi

if [ ! -f "/data/machine-id" ]; then
  cp "/etc/machine-id" "/data"
fi
```

Untuk mempermudah proses penambahan file-file tersebut, buatlah sebuah file bitbake recipe bernama voice-ui-framework_1.0.bbappend yang berfungsi untuk memberikan deklarasi tambahan ke dalam file bitbake recipe bernama voice-ui-framework_1.0.bb yang sudah ada sebelumnya. Hal ini bertujuan untuk menghindari perlunya perubahan konfigurasi layer yocto.

Isi file voice-ui-framework_1.0.bbappend adalah sebagai berikut:

```
SRC_URI +=  "\
            file://bk-pairingapp.service \
            file://bk-getdataapi.service \
            file://bk-sound \
            file://bk-voiceui.sh \
            file://bk-voiceui.service \
            "

#Specify install appends
do_install_append() {
    install -d ${D}${userfsdatadir}
    cp -Rf ${WORKDIR}/bk-sound/* ${D}${userfsdatadir}

    install -d ${D}${bindir}
    install -m 0744 ${WORKDIR}/bk-voiceui.sh ${D}/${bindir}

    install -d ${D}${systemd_system_unitdir}
    install -m 0644 ${WORKDIR}/bk-pairingapp.service -D ${D}${systemd_system_unitdir}
    install -m 0644 ${WORKDIR}/bk-getdataapi.service -D ${D}${systemd_system_unitdir}
    install -m 0644 ${WORKDIR}/bk-voiceui.service -D ${D}${systemd_system_unitdir}
}

SYSTEMD_SERVICE_${PN} += "bk-pairingapp.service \
bk-getdataapi.service \
bk-voiceui.service"
```

Pada realitanya, aplikasi bahasakita-voiceui perlu diintegrasikan dengan aplikasi wakeword engine. Apabila menggunakan aplikasi wakeword engine Mycroft Precise, isi file voice-ui-framework_1.0.bbappend perlu tambahan sebagai berikut:

```
SRC_URI +=  "\
            file://bk-wakeword.service \
            file://precise-engine_0.4.0_aarch64.tar.gz \
            file://mycroft \
            "

#Specify install appends
do_install_append() {
    install -d ${D}${userfsdatadir}
    cp -Rf ${WORKDIR}/mycroft ${D}${userfsdatadir}

    install -d ${D}${libdir}
    cp -Rf ${WORKDIR}/precise-engine ${D}${libdir}

    install -d ${D}${systemd_system_unitdir}
    install -m 0644 ${WORKDIR}/bk-wakeword.service -D ${D}${systemd_system_unitdir}
}

SYSTEMD_SERVICE_${PN} += "bk-wakeword.service"
```

Proses integrasi aplikasi wakeword engine Mycroft Precise dengan aplikasi Bahasa Kita beserta file-file yang diperlukan akan disampaikan pada bagian selanjutnya dari dokumen ini.

Menambahkan Aplikasi Mycroft Precise ke dalam Image QCS404

Sebelum menambahkan aplikasi Mycroft Precise, pengguna harus menambahkan aplikasi Precise Engine terlebih dahulu. Precise Engine dibuat menggunakan bahasa pemrograman Python sehingga aplikasi ini hanya bisa ditambahkan sebagai aplikasi dalam bentuk file executable terkompresi. Ada 2 alternatif untuk memperoleh file executable terkompresi, yaitu:

1. Menggunakan file executable terkompresi yang sudah dibuat dan diuji.
   File executable terkompresi ini biasanya sudah dibuat berdasarkan standar fitur tertentu dan sudah teruji fungsionalitasnya. File ini direkomendasikan apabila tidak ada perubahan fitur yang diperlukan atau ketika standar fitur sudah sesuai dengan kebutuhan. File executable terkompresi aplikasi Precise Engine dapat diperoleh dari repositori Prasimax.
2. Membuat file executable terkompresi dari source code.
   Membuat file executable akan selalu diperlukan apabila terjadi perubahan baik penambahan atau pengurangan fitur. File executable yang dihasilkan perlu melalui berbagai proses pengujian sebelum bisa digunakan dan dikemas sebagai file executable terkompresi. Apabila pengguna ingin membuat file executable terkompresi, pengguna dapat mengikuti panduan dalam dokumen berikut:

   SSD06_Precise_Engine.pdf : Membuat File Aplikasi Precise Engine untuk Mycroft Precise pada Ubuntu 16.04 LTS

   Sebelum mengikuti panduan pada dokumen tersebut, pengguna harus mengikuti panduan dalam dokumen berikut:

   SSD02_Qemu_Ubuntu_Install.pdf : Memasang Qemu Versi Terbaru pada OS Ubuntu 16.04 LTS
   SSD03_Qemu_Ubuntu_Arm.pdf : Memasang Ubuntu Arm (32bit) pada Qemu Versi Terbaru
   SSD04_Qemu_Ubuntu_Aarch64.pdf : Memasang Ubuntu Aarch64 (64bit) pada Qemu Versi Terbaru
   SSD05_PortAudio_Pulse.pdf : Memasang Library PortAudio dengan Low Level PulseAudio pada Ubuntu 16.04 LTS

   Perhatikan bahwa Ubuntu Arm digunakan untuk mengemulasikan sistem nf-32 dan Ubuntu Aarch64 digunakan untuk mengemulasikan sistem nf-64 pada image QCS404. Untuk menambahkan aplikasi dalam bentuk file executable terkompresi, buatlah sebuah file bitbake recipe bernama voice-ui-framework_1.0.bbappend yang berfungsi untuk memberikan deklarasi tambahan ke dalam file bitbake recipe bernama voice-ui-framework_1.0.bb yang sudah ada sebelumnya.

   Isi file voice-ui-framework_1.0.bbappend adalah sebagai berikut:

```
SRC_URI +=  "\
            file://precise-engine_0.4.0_aarch64.tar.gz \
            "

#Specify install appends
do_install_append() {
    install -d ${D}${libdir}
    cp -Rf ${WORKDIR}/precise-engine ${D}${libdir}
}
```

   Perhatikan bahwa precise-engine_0.4.0_aarch64.tar.gz adalah hasil build dari Ubuntu Aarch64 sedangkan hasil build dari Ubuntu Arm akan bernama precise-engine_0.4.0_armv7l.tar.gz dimana perbedaan nama tersebut menunjukkan perbedaan arsitektur sistem.

Aplikasi Precise Engine memerlukan Precise Runner supaya dapat berfungsi. Precise Runner adalah paket library Python yang akan menjalankan aplikasi Precise Engine bersama dengan model wakeword. Ada 2 alternatif dalam menggunakan Precise Runner, yaitu:

1. Memasang Python dan seluruh paket library Python terkait Mycroft Precise Alternatif ini digunakan apabila aplikasi masih dalam tahap pengembangan seperti integrasi aplikasi Mycroft Precise dengan aplikasi Bahasa Kita. Untuk memasang paket library Python, buatlah file bitbake recipe untuk masing-masing paket library. Seluruh paket library Python terkait Mycroft Precise yang dimaksud adalah Precise Runner (python3-precise-runner_0.3.1.bb) dan PyAudio (python3-pyaudio_0.2.11.bb). Paket library Python PyAudio memerlukan library PortAudio (portaudio-pulse) untuk bisa berfungsi sehingga buatlah file bitbake recipe bernama portaudio-pulse_20161030.bb untuk memasang library tersebut.

   Isi file python3-precise-runner_0.3.1.bb adalah sebagai berikut:

```
SUMMARY = "A lightweight, simple-to-use, RNN wake word listener."
HOMEPAGE = "https://github.com/MycroftAI/mycroft-precise"
LICENSE = "Apache-2.0"
LIC_FILES_CHKSUM = "file://README.md;md5=2ad29e930f3107d52d2a55728bf62116"

SRC_URI[md5sum] = "a2434be110444192e804f4dada0ccecf"
SRC_URI[sha256sum] =
"1a464209fb4bf0a3f5d5a428310cb2a70487a01a6bc3a960d1dda90af896b80d"

inherit pypi setuptools3
```

   Isi file python3-pyaudio_0.2.11.bb adalah sebagai berikut:

```
SUMMARY = "PyAudio provides Python bindings for PortAudio, the cross-platform audio
I/O library"
SECTION = "devel/python"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://README;md5=288793c2b9b05bd67abbd2a8f5d144f7"

inherit pypi setuptools3

PYPI_PACKAGE = "PyAudio"

SRC_URI[md5sum] = "7e4c88139284033f67b4336c74eda3b8"
SRC_URI[sha256sum] =
"93bfde30e0b64e63a46f2fd77e85c41fd51182a4a3413d9edfaf9ffaa26efb74"

DEPENDS += "portaudio-pulse"

RDEPENDS_${PN} += "portaudio-pulse"
```

   Isi file portaudio-pulse_20161030.bb adalah sebagai berikut:

```
SUMMARY = "A portable audio library"
SECTION = "libs/multimedia"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://LICENSE.txt;md5=26107732c2ab637c5710446fcfaf02df"

PV = "v190600"
SRC_URI = "file://portaudio-pulse.tar.gz"
SRC_URI[md5sum] = "4df8224e047529ca9ad42f0521bf81a8"
SRC_URI[sha256sum] =
"f5a21d7dcd6ee84397446fa1fa1a0675bb2e8a4a6dceb4305a8404698d8d1513"

S = "${WORKDIR}/portaudio-pulse"

inherit autotools pkgconfig texinfo

PACKAGECONFIG ??= "pulseaudio"
PACKAGECONFIG[pulseaudio] = "--with-pulseaudio, --without-pulseaudio, pulseaudio,"

EXTRA_OECONF = "--with-pulseaudio --without-alsa --without-oss --without-asihpi --
without-jack"
EXTRA_OEMAKE = "LIBTOOL='${B}/*-libtool --tag=CC'"
```

Perhatikan bahwa portaudio-pulse.tar.gz adalah file hasil kompresi source code PortAudio dengan Low Level PulseAudio. Untuk mengintegrasikan aplikasi Mycroft Precise dengan aplikasi Bahasa Kita, diperlukan 1 file script python bernama bk-wakeword.py yang berfungsi untuk menjalankan aplikasi Precise Engine bersama dengan model wakeword. Model wakeword ini terdiri dari 2 file bernama hey-mycroft.tflite.params dan hey-mycroft.tflite. Ketiga file ini kemudian disimpan ke dalam 1 folder bernama mycroft. Untuk memulai script bk-wakeword.py, diperlukan 1 file service bernama bk-wakeword.service.

Isi file bk-wakeword.py adalah sebagai berikut:

```
from precise_runner import PreciseEngine, PreciseRunner
from time import sleep
from subprocess import call

def on_act():
    # print('wake word detected')
    call(["posix-client", "kws_active"])

# initiate precise engine with mycroft model
engine = PreciseEngine('/usr/lib64/precise-engine/precise-engine',
'/data/mycroft/hey-mycroft.tflite')

# initiate precise runner that will listen, predict, and detect wakeword
runner = PreciseRunner(engine, on_activation=on_act, trigger_level=8,
sensitivity=0.5)

# start runner
# print('Initialize Engine')
runner.start()

# keep main thread active until user interrupt
try:
    while 1:
        # print('listening...')
        sleep(60)
except:
    runner.stop()
```

Isi file bk-wakeword.service adalah sebagai berikut:

```
[Unit]
Description=Bahasa Kita Wakeword Service
Requires=bk-voiceui.service
After=bk-voiceui.service

[Service]
Restart=on-failure
RestartSec=10
EnvironmentFile=/tmp/dbus-session
ExecStart=/usr/bin/python3 /data/mycroft/bk-wakeword.py

[Install]
WantedBy=bk-voiceui.service
```

Ubah isi file voice-ui-framework_1.0.bbappend menjadi sebagai berikut:

```
SRC_URI +=  "\
            file://bk-pairingapp.service \
            file://bk-getdataapi.service \
            file://bk-sound \
            file://bk-voiceui.sh \
            file://bk-voiceui.service \
            file://bk-wakeword.service \
            file://precise-engine_0.4.0_aarch64.tar.gz \
            file://mycroft \
            "
```

```
#Specify install appends
do_install_append() {
    install -d ${D}${userfsdatadir}
    cp -Rf ${WORKDIR}/bk-sound/* ${D}${userfsdatadir}
    cp -Rf ${WORKDIR}/mycroft ${D}${userfsdatadir}

    install -d ${D}${libdir}
    cp -Rf ${WORKDIR}/precise-engine ${D}${libdir}

    install -d ${D}${bindir}
    install -m 0744 ${WORKDIR}/bk-voiceui.sh ${D}/${bindir}

    install -d ${D}${systemd_system_unitdir}
    install -m 0644 ${WORKDIR}/bk-pairingapp.service -D ${D}${systemd_system_unitdir}
    install -m 0644 ${WORKDIR}/bk-getdataapi.service -D ${D}${systemd_system_unitdir}
    install -m 0644 ${WORKDIR}/bk-voiceui.service -D ${D}${systemd_system_unitdir}
    install -m 0644 ${WORKDIR}/bk-wakeword.service -D ${D}${systemd_system_unitdir}
}

SYSTEMD_SERVICE_${PN} += "bk-pairingapp.service \
bk-getdataapi.service \
bk-voiceui.service \
bk-wakeword.service"
```

Tambahkan baris berikut ke dalam file local.conf:

```
IMAGE_INSTALL_append += "python3-core python3-precise-runner python3-pyaudio portaudio-pulse"
```

Apabila semua sudah selesai, build ulang image. Ada beberapa file bitbake recipe yang ditambahkan untuk proses pengembangan aplikasi Bahasa Kita tetapi tidak disertakan dalam dokumen ini. Cek repository Prasimax untuk mempelajari file-file tersebut.

2. Membuat aplikasi Mycroft Precise dalam bentuk file executable terkompresi

Untuk membuat aplikasi Mycroft Precise dalam bentuk file executable terkompresi, ikuti panduan dalam dokumen berikut:

SSD07_Mycroft_Precise.pdf : Membuat File Aplikasi Mycroft Precise pada Ubuntu 16.04 LTS

Apabila build aplikasi sudah berhasil diselesaikan, pastikan untuk menguji aplikasi terlebih dahulu sebelum membuat file terkompresi. Untuk menjalankan aplikasi Mycroft Precise dalam bentuk file terkompresi, diperlukan file konfigurasi dalam format json bernama mycroft-precise.json dengan isi sebagai berikut:

```
{"engine_path": "/data/python_lib/precise-engine", "model_path": "/data/hey-mycroft.tflite", "trigger_level": 8, "sensitivity": 0.5, "command": ["adk-message-send", "led_start_pattern{pattern:7}"]}
```

Perhatikan bahwa engine_path adalah lokasi aplikasi Precise Engine, model_path adalah lokasi model wakeword, trigger_level dan sensitivity adalah parameter untuk aplikasi Precise Engine, dan command adalah parameter perintah yang harus dieksekusi saat wakeword terdeteksi dengan format sesuai parameter fungsi call dalam paket library Python subprocess. Gunakan list kosong jika tidak ingin menggunakan command. Untuk mempermudah proses development, buatlah 1 folder bernama python_lib kemudian salin seluruh file executable hasil build aplikasi ke dalam folder tersebut. Setelah itu, buatlah file terkompresi dengan format penamaan yang dapat menunjukkan arsitektur sistem build seperti python_lib_aarch64.tar.gz untuk Ubuntu Aarch64 (sesuai sistem nf-64) dan python_lib_armv7l.tar.gz untuk Ubuntu Arm (sesuai sistem nf-32).

Untuk menambahkan aplikasi Mycroft Precise dengan alternatif ini, buatlah file voice-ui-framework_1.0.bbappend dengan isi sebagai berikut:

```
SRC_URI +=  "\
            file://python_lib_aarch64.tar.gz \
            "

#Specify install appends
do_install_append() {
    install -d ${D}${libdir}
    cp -Rf ${WORKDIR}/python_lib ${D}${libdir}
}
```

Perhatikan bahwa python_lib_aarch64.tar.gz adalah file aplikasi terkompresi untuk sistem nf-64 yang harus diganti dengan python_lib_armv7l.tar.gz untuk sistem nf-32. Pada alternatif ini, paket library Python terkait Mycroft Precise seperti Precise Runner (python3-precise-runner_0.3.1.bb) dan PyAudio (python3-pyaudio_0.2.11.bb) sudah tidak diperlukan tetapi portaudio-pulse_20161030.bb masih diperlukan supaya aplikasi dapat berjalan. Selain itu, script bk-wakeword.py juga tentunya sudah tidak diperlukan.

Apabila diperlukan, beberapa file executable hasil build (dari precise-engine dan mycroft-precise) dapat digabungkan ke dalam satu folder python_lib untuk menghilangkan redundancy file dan mengurangi penggunaan memory.

Untuk mengintegrasikan aplikasi Mycroft Precise dengan aplikasi Bahasa Kita, sesuaikan file bk-wakeword.service dan file voice-ui-framework_1.0.bbappend yang telah dibuat sebelumnya.

Ubah isi file bk-wakeword.service menjadi sebagai berikut:

```
[Unit]
Description=Bahasa Kita Wakeword Service
Requires=bk-voiceui.service
After=bk-voiceui.service

[Service]
Restart=on-failure
RestartSec=10
EnvironmentFile=/tmp/dbus-session
ConditionPathExists=/data/mycroft-precise.json
ExecStart=/usr/lib64/python_lib/mycroft-precise

[Install]
WantedBy=bk-voiceui.service
```

Ubah isi file voice-ui-framework_1.0.bbappend menjadi sebagai berikut:

```
SRC_URI +=  "\
            file://bk-pairingapp.service \
            file://bk-getdataapi.service \
            file://bk-sound \
            file://bk-voiceui.sh \
            file://bk-voiceui.service \
            file://bk-wakeword.service \
            file://precise-engine_0.4.0_aarch64.tar.gz \
            file://python_lib_aarch64.tar.gz \
            file://mycroft \
            "

#Specify install appends
do_install_append() {
    install -d ${D}${userfsdatadir}
    cp -Rf ${WORKDIR}/bk-sound/* ${D}${userfsdatadir}
    cp -Rf ${WORKDIR}/mycroft ${D}${userfsdatadir}
```

```
    install -d ${D}${libdir}
    cp -Rf ${WORKDIR}/precise-engine ${D}${libdir}
    cp -Rf ${WORKDIR}/python_lib ${D}${libdir}

    install -d ${D}${bindir}
    install -m 0744 ${WORKDIR}/bk-voiceui.sh ${D}/${bindir}

    install -d ${D}${systemd_system_unitdir}
    install -m 0644 ${WORKDIR}/bk-pairingapp.service -D ${D}${systemd_system_unitdir}
    install -m 0644 ${WORKDIR}/bk-getdataapi.service -D ${D}${systemd_system_unitdir}
    install -m 0644 ${WORKDIR}/bk-voiceui.service -D ${D}${systemd_system_unitdir}
    install -m 0644 ${WORKDIR}/bk-wakeword.service -D ${D}${systemd_system_unitdir}
}

SYSTEMD_SERVICE_${PN} += "bk-pairingapp.service \
bk-getdataapi.service \
bk-voiceui.service \
bk-wakeword.service"
```

Hapus baris berikut dari dalam file local.conf apabila tidak diperlukan:

```
IMAGE_INSTALL_append += "python3-core python3-precise-runner python3-pyaudio portaudio-pulse"
```

Apabila semua sudah selesai, build ulang image. Pastikan untuk membuat file mycroft-precise.json sebelum menjalankan aplikasi Mycroft Precise. Cek repository Prasimax untuk mempelajari file-file yang telah dibahas.