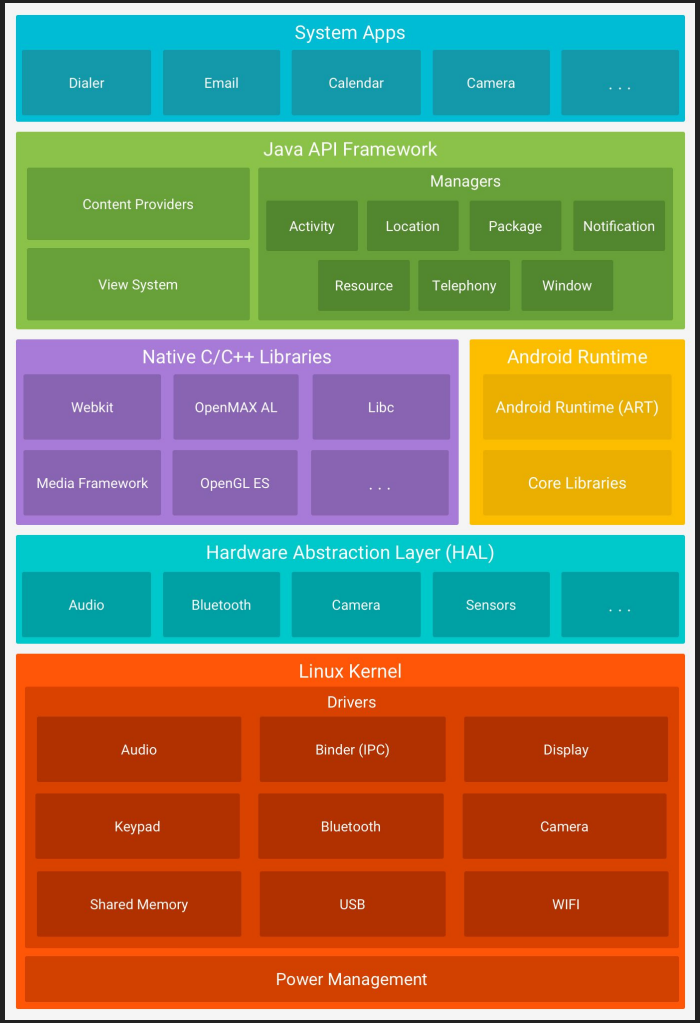
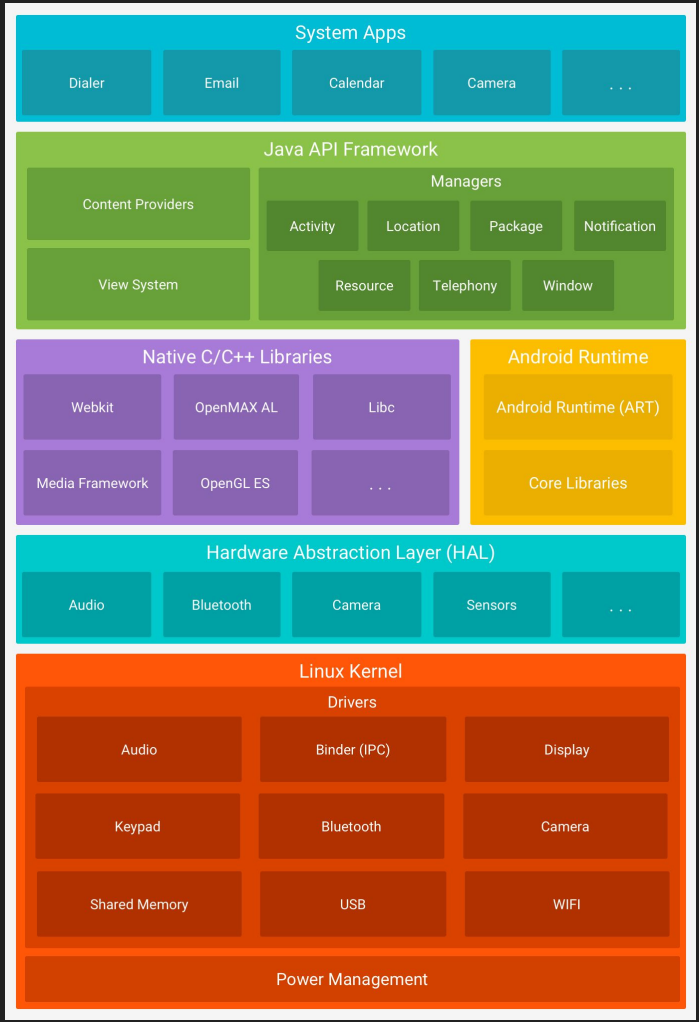
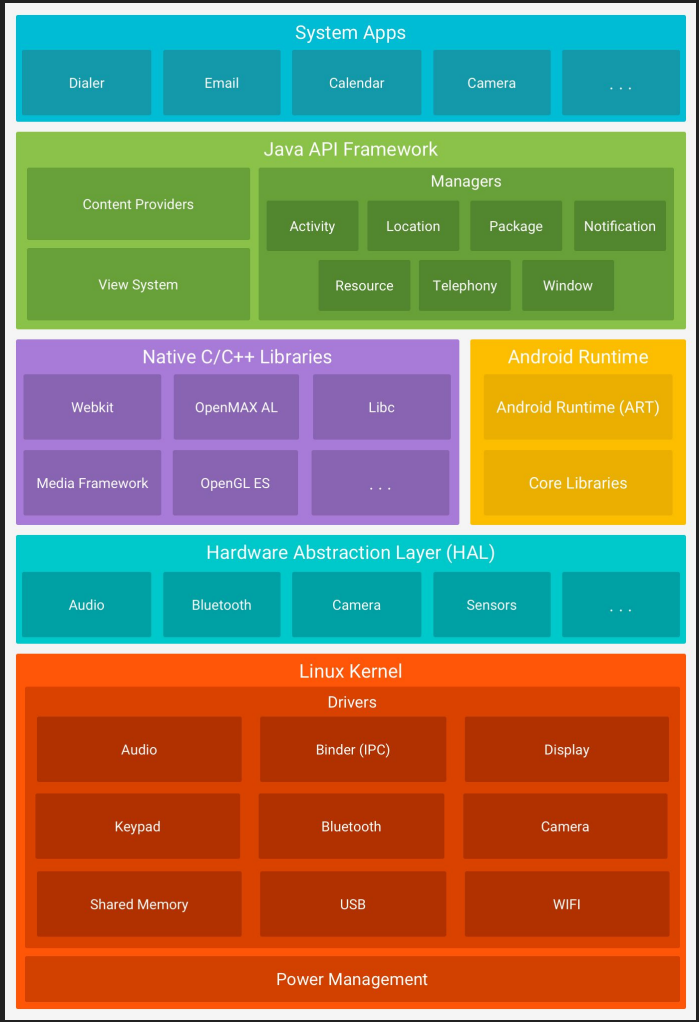
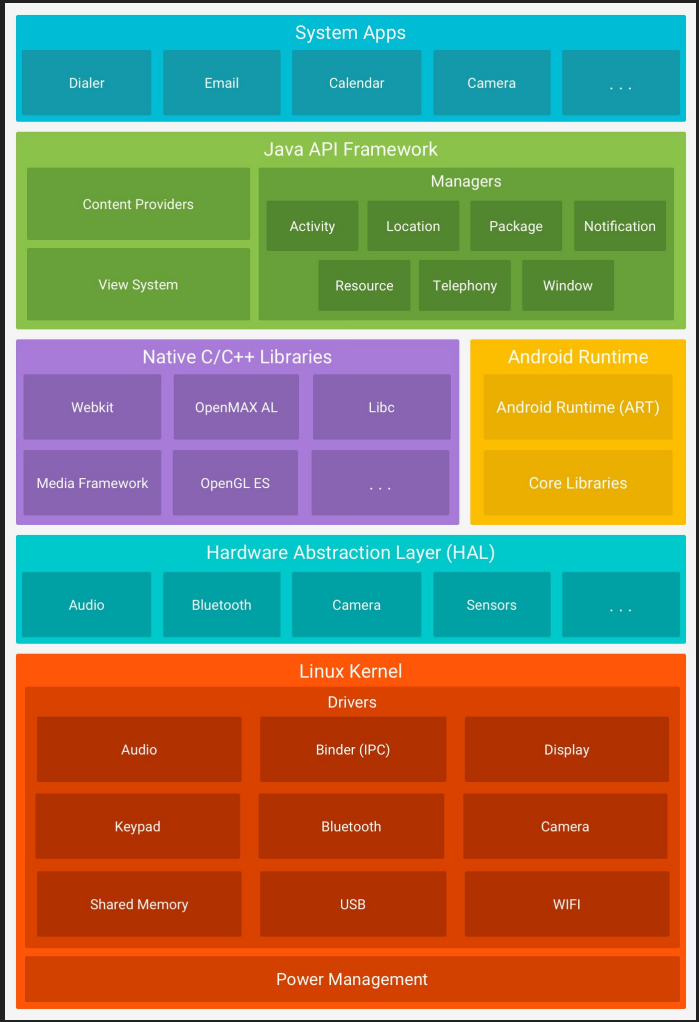
Huins System Service

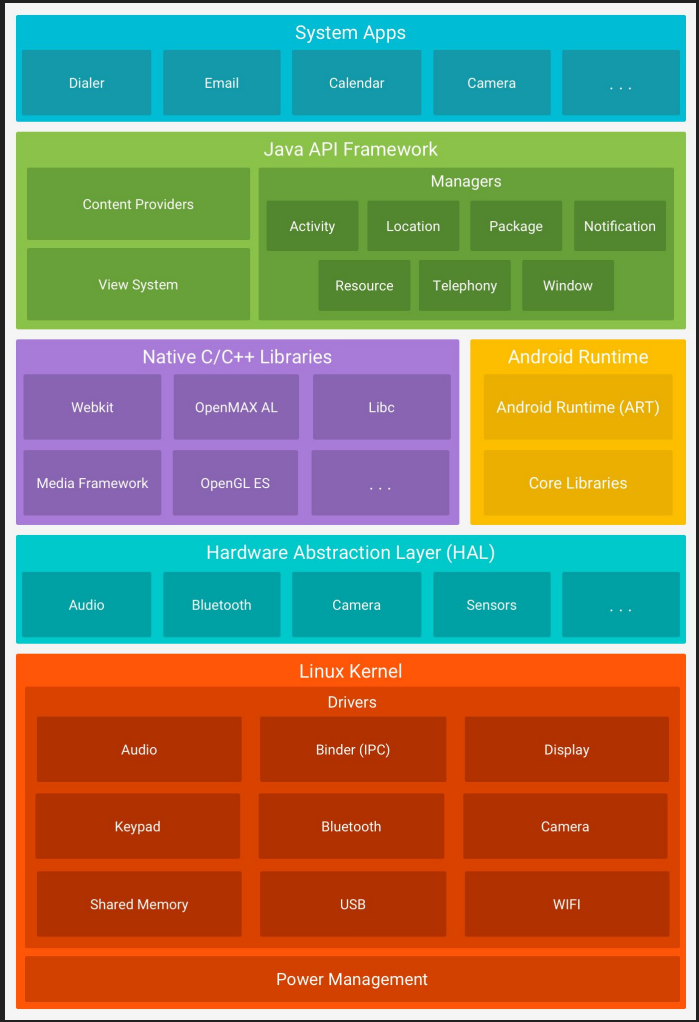
20151543 민지우
20151623 한상구

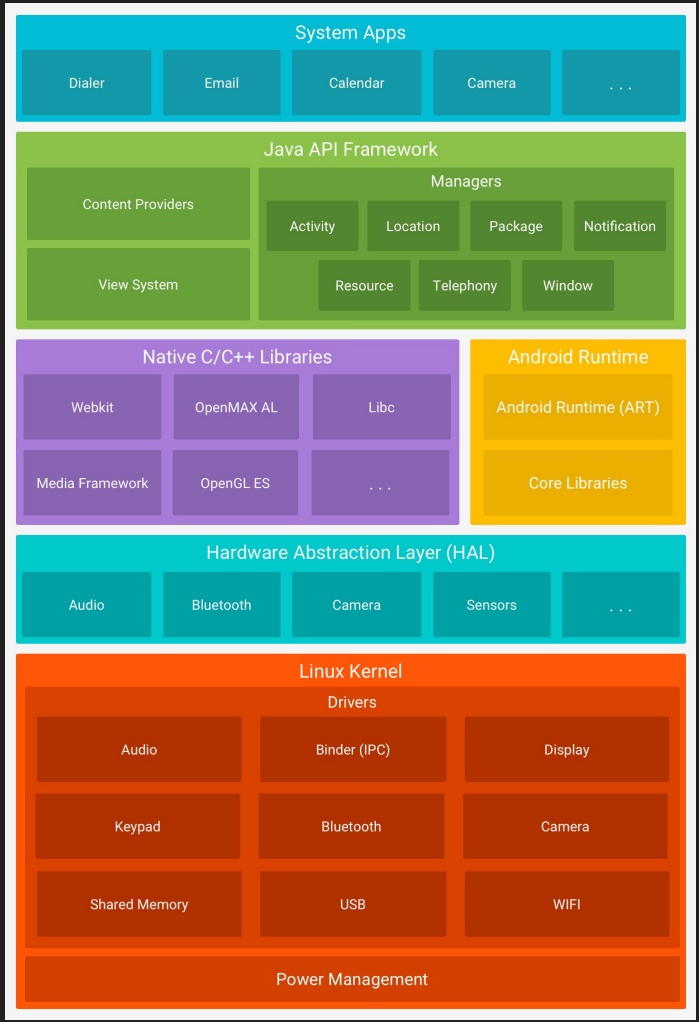

System Apps
Dialer
Email
Calendar
Camera
. . .
Java API Framework
Content Providers
View System
Managers
Activity
Location
Package
Notification
Resource
Telephony
Window
Native C/C++ Libraries
Webkit
OpenMAX AL
Libc
Media Framework
OpenGL ES
. . .
Android Runtime
Android Runtime (ART)
Core Libraries
Hardware Abstraction Layer (HAL)
Audio
Bluetooth
Camera
Sensors
. . .
Linux Kernel
Drivers
Audio
Binder (IPC)
Display
Keypad
Bluetooth
Camera
Shared Memory
USB
WIFI
Power Management

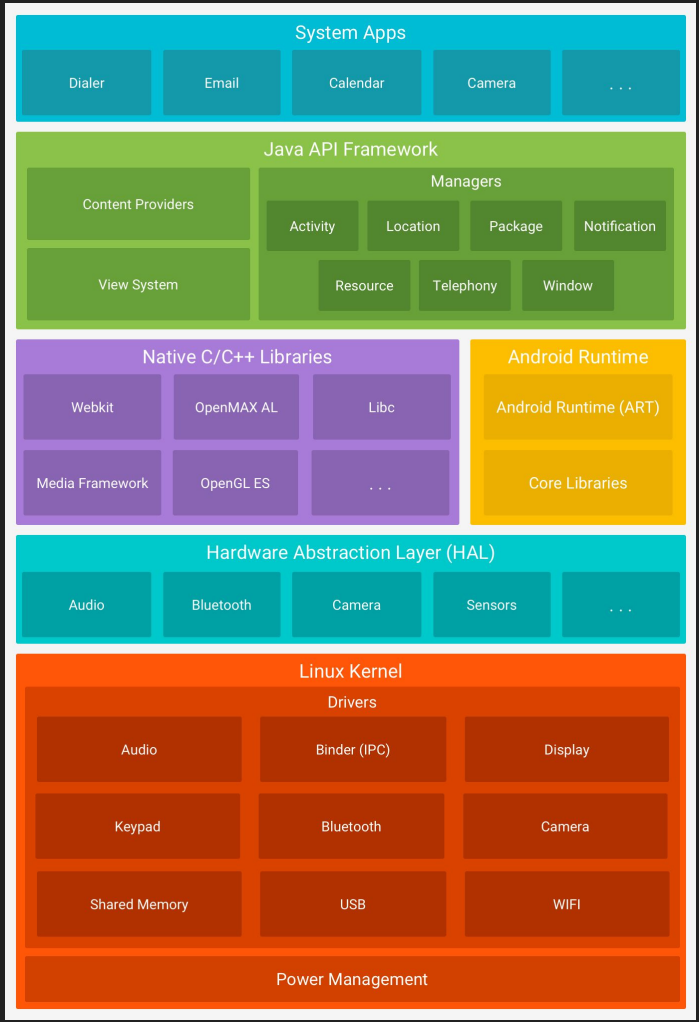

System Apps
Dialer
Email
Calendar
Camera
. . .
Java API Framework
Content Providers
View System
Managers
Activity
Location
Package
Notification
Resource
Telephony
Window
Native C/C++ Libraries
Webkit
OpenMAX AL
Libc
Media Framework
OpenGL ES
. . .
Android Runtime
Android Runtime (ART)
Core Libraries
Hardware Abstraction Layer (HAL)
Audio
Bluetooth
Camera
Sensors
. . .
Linux Kernel
Drivers
Audio
Binder (IPC)
Display
Keypad
Bluetooth
Camera
Shared Memory
USB
WIFI
Power Management


System Apps
Dialer
Email
Calendar
Camera
. . .
Java API Framework
Content Providers
View System
Managers
Activity
Location
Package
Notification
Resource
Telephony
Window
Native C/C++ Libraries
Webkit
OpenMAX AL
Libc
Media Framework
OpenGL ES
. . .
Android Runtime
Android Runtime (ART)
Core Libraries
Hardware Abstraction Layer (HAL)
Audio
Bluetooth
Camera
Sensors
. . .
Linux Kernel
Drivers
Audio
Binder (IPC)
Display
Keypad
Bluetooth
Camera
Shared Memory
USB
WIFI
Power Management


System Apps
Dialer
Email
Calendar
Camera
. . .
Java API Framework
Content Providers
View System
Managers
Activity
Location
Package
Notification
Resource
Telephony
Window
Native C/C++ Libraries
Webkit
OpenMAX AL
Libc
Media Framework
OpenGL ES
. . .
Android Runtime
Android Runtime (ART)
Core Libraries
Hardware Abstraction Layer (HAL)
Audio
Bluetooth
Camera
Sensors
. . .
Linux Kernel
Drivers
Audio
Binder (IPC)
Display
Keypad
Bluetooth
Camera
Shared Memory
USB
WIFI
Power Management


System Apps
Dialer
Email
Calendar
Camera
. . .
Java API Framework
Content Providers
View System
Managers
Activity
Location
Package
Notification
Resource
Telephony
Window
Native C/C++ Libraries
Webkit
OpenMAX AL
Libc
Media Framework
OpenGL ES
. . .
Android Runtime
Android Runtime (ART)
Core Libraries
Hardware Abstraction Layer (HAL)
Audio
Bluetooth
Camera
Sensors
. . .
Linux Kernel
Drivers
Audio
Binder (IPC)
Display
Keypad
Bluetooth
Camera
Shared Memory
USB
WIFI
Power Management


System Apps
Dialer
Email
Calendar
Camera
. . .
Java API Framework
Content Providers
View System
Managers
Activity
Location
Package
Notification
Resource
Telephony
Window
Native C/C++ Libraries
Webkit
OpenMAX AL
Libc
Media Framework
OpenGL ES
. . .
Android Runtime
Android Runtime (ART)
Core Libraries
Hardware Abstraction Layer (HAL)
Audio
Bluetooth
Camera
Sensors
. . .
Linux Kernel
Drivers
Audio
Binder (IPC)
Display
Keypad
Bluetooth
Camera
Shared Memory
USB
WIFI
Power Management


System Apps
Dialer
Email
Calendar
Camera
. . .
Java API Framework
Content Providers
View System
Managers
Activity
Location
Package
Notification
Resource
Telephony
Window
Native C/C++ Libraries
Webkit
OpenMAX AL
Libc
Media Framework
OpenGL ES
. . .
Android Runtime
Android Runtime (ART)
Core Libraries
Hardware Abstraction Layer (HAL)
Audio
Bluetooth
Camera
Sensors
. . .
Linux Kernel
Drivers
Audio
Binder (IPC)
Display
Keypad
Bluetooth
Camera
Shared Memory
USB
WIFI
Power Management

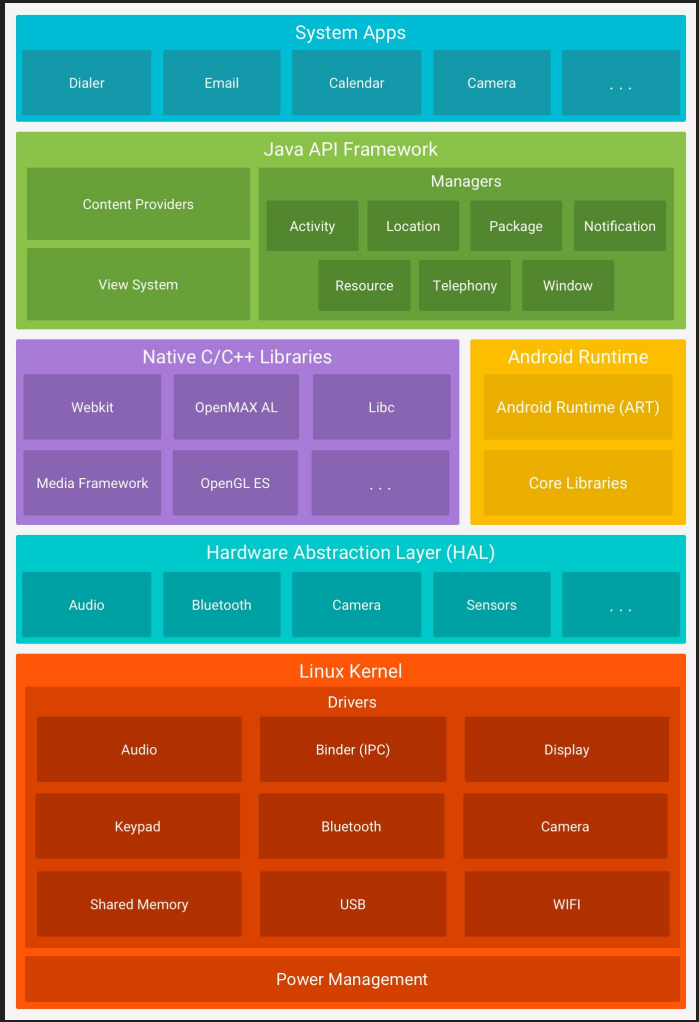
```
embe@embe-PNVKB0A0-Samsung-DeskTop:~/cse4116_project/android$ tree
.
├── frameworks
│   └── base
│       ├── Android.mk
│       ├── core
│       │   └── java
│       │       └── android
│       │           └── os
│       │               └── IhuinsService.aidl
│       └── services
│           ├── java
│           │   └── com
│           │       └── android
│           │           └── server
│           │               └── HuinsService.java
│           └── jni
│               ├── Android.mk
│               ├── com_android_server_HuinsService.cpp
│               └── onload.cpp
└── hardware
    ├── imx
    │   └── mx6
    │       └── libgralloc_wrapper
    │           └── framebuffer.cpp
    └── libhardware
        └── include
            └── hardware
                ├── Android.mk
                ├── huins.c
                └── huins.h
```

```c
static int open_huins(const struct hw_module_t *module,
                char const *name,
                struct hw_device_t **device)
{
    UNUSED(name);

    struct huins_device_t *dev = calloc(1, sizeof(struct huins_device_t));
    if (!dev)
            return -ENOMEM;

    dev->common.tag = HARDWARE_DEVICE_TAG;
    dev->common.version = 0;
    dev->common.module = (struct hw_module_t*)module;
    dev->common.close = (int (*)(struct hw_device_t*))close_huins;

    dev->set_dot_matrix = huins_set_dot_matrix;
    dev->set_fnd = huins_set_fnd;
    dev->set_lcd = huins_set_fcd;
    dev->set_led = huins_set_led;
    dev->set_buzzer = huins_set_buzzer;
    dev->set_motor = huins_set_motor;

    *device = (struct hw_device_t*)dev;
    puts("HAL for huins board initializes");

    return 0;
}

static struct hw_module_methods_t huins_module_methods = {
    .open = open_huins,
};

struct huins_module_t HAL_MODULE_INFO_SYM = {
    .common = {
        .tag = HARDWARE_MODULE_TAG,
        .module_api_version = HUINS_API_VERSION_0_1,
        .hal_api_version = HARDWARE_HAL_API_VERSION,
        .id = HUINS_MODULE_ID,
        .name = "HAL for control huins board",
        .author = "Sanggu Han",
        .methods = &huins_module_methods,
    }
};
```

generate device object

set functions

give address of hardware object


System Apps
Dialer
Email
Calendar
Camera
. . .
Java API Framework
Content Providers
View System
Managers
Activity
Location
Package
Notification
Resource
Telephony
Window
Native C/C++ Libraries
Webkit
OpenMAX AL
Libc
Media Framework
OpenGL ES
. . .
Android Runtime
Android Runtime (ART)
Core Libraries
Hardware Abstraction Layer (HAL)
Audio
Bluetooth
Camera
Sensors
. . .
Linux Kernel
Drivers
Audio
Binder (IPC)
Display
Keypad
Bluetooth
Camera
Shared Memory
USB
WIFI
Power Management

```
embe@embe-PNVKB0A0-Samsung-DeskTop:~/cse4116_project/android$ tree
.
├── frameworks
│   └── base
│       ├── Android.mk
│       ├── core
│       │   └── java
│       │       └── android
│       │           └── os
│       │               └── IhuinsService.aidl
│       └── services
│           ├── java
│           │   └── com
│           │       └── android
│           │           └── server
│           │               └── HuinsService.java
│           └── jni
│               ├── Android.mk
│               ├── com_android_server_HuinsService.cpp
│               └── onload.cpp
└── hardware
    ├── imx
    │   └── mx6
    │       └── libgralloc_wrapper
    │           └── framebuffer.cpp
    └── libhardware
        └── include
            └── hardware
                ├── Android.mk
                ├── huins.c
                └── huins.h
```

```
#define LOG_TAG "huins"

#include "jni.h"
#include "JNIHelp.h"
#include "androidd_runtime/AndroidRuntime.h"

#include <utils/misc.h>
#include <utils/log.h>
#include <hardware/hardware.h>
#include <hardware/huins.h>

#include <stdio.h>


namespace android
{

    huins_device_t *huins_dev = NULL;


    static jlong init_native(JNIEnv *env, jobject clazz)
    {
        int err;
        huins_module_t *module;

        err = hw_get_module(HUINS_MODULE_ID, (hw_module_t const**)&module);
        if(err == 0)
        {
            if(module->common.methods->open((hw_module_t*)module, "", ((hw_device_t**)&huins_dev)) !
= 0)
                return 0;
        }

        long tmp = reinterpret_cast<long>(huins_dev);

        return tmp;
    }

    static void finalize_native(JNIEnv *env, jobject clazz, jlong ptr)
    {
        huins_device_t *dev = reinterpret_cast<huins_device_t*>(ptr);
        if(dev == NULL)
            return;

        free(dev);
    }

    static void set_dot_matrix_native(JNIEnv *env, jobject clazz, jint m)
    {
        if(huins_dev == NULL)
            return;

        huins_dev->set_dot_matrix(m);
    }

    static void set_fnd_native(JNIEnv *env, jobject clazz, jint n)
    {
        if(huins_dev == NULL)
            return;
```

Use HAL

get hardware object

get device address via open()

```
static JNINativeMethod method_table[] =
{
    {"init_native", "()J", (void*)init_native},
    {"finalize_native", "(J)V", (void*)finalize_native},
    {"set_dot_matrix_native", "(I)V", (void *)set_dot_matrix_native},
    {"set_fnd_native", "(I)V", (void *)set_fnd_native},
    {"set_lcd_native", "([B)V", (void *)set_lcd_native},
    {"set_led_native", "(I)V", (void *)set_led_native},
    {"set_buzzer_native", "(I)V", (void *)set_buzzer_native},
    {"set_motor_native", "(III)V", (void *)set_motor_native},
};

int register_android_server_HuinsService(JNIEnv *env)
{
    return jniRegisterNativeMethods(env, "com/android/server/HuinsService",
                                    method_table, NELEM(method_table));
}
```

register those methods

```java
import android.os.Looper;
import android.os.Message;
import android.os.Process;
import android.os.IhuinsService;
import android.util.Slog;

public class HuinsService extends IhuinsService.Stub
{

    private static final String TAG = "HuinsService";
    private Context mContext;
    private int mNativePointer;

    HuinsService(Context context){
        mContext = context;
        init_native();
    }


    @Override
    public void set_dot_matrix(int m) {
        set_dot_matrix_native(m);
    }

    @Override
    public void set_fnd(int n) {
        set_fnd_native(n);
    }

    @Override
    public void set_lcd(byte[] buf) {
        set_lcd_native(buf);
    }

    @Override
    public void set_led(int bm) {
        set_led_native(bm);
    }

    @Override
    public void set_buzzer(int buzz) {
        set_buzzer_native(buzz);
    }

    @Override
    public void set_motor(int action, int direction, int speed) {
        set_motor_native(action, direction, speed);
    }

    private static native long init_native();
    private static native void finalize_native(long ptr);
    private static native void set_dot_matrix_native(int m);
    private static native void set_fnd_native(int n);
    private static native void set_lcd_native(byte[] buf);
    private static native void set_led_native(int bm);
    private static native void set_buzzer_native(int buzz);
    private static native void set_motor_native(int action, int direction, int
    speed);
```

interface using jni

native methods

```
.
└── frameworks
    └── base
        ├── Android.mk
        ├── core
        │   └── java
        │       └── android
        │           ├── app
        │           │   └── ContextImpl.java
        │           ├── content
        │           │   └── Context.java
        │           └── os
        │               ├── HuinsSystemManager.java
        │               └── IHuinsSystemService.aidl
        └── services
            ├── java
            │   └── com
            │       └── android
            │           └── server
            │               ├── huins
            │               │   ├── HuinsInputSender.java
            │               │   ├── HuinsSystemController.java
            │               │   └── HuinsSystemServiceImpl.java
            │               └── SystemServer.java
            └── jni
                ├── Android.mk
                ├── com_android_server_huins_HuinsSystemController.cpp
                ├── com_android_server_huins_HuinsSystemController.h
                └── onload.cpp
```

4: register to context

3: client interface

2: system service

1 :native jni

```
    }

    static void writeMOTOR_native(
            JNIEnv *env,
            jobject thiz,
            jint fd,
            jint action,
            jint direction,
            jint speed
            ) {
        int op[3];
        op[0] = action;
        op[1] = direction;
        op[2] = speed;

        ioctl(fd, IOCTL_SET_MOTOR, op);
    }

    static void endInputDevices_native(
            JNIEnv *env,
            jobject thiz,
            jint fd
    ){
        close(fd);
    }

    static void endOutputDevices_native(
            JNIEnv *env,
            jobject thiz,
            jint fd
            ){
        close(fd);
    }

//-----------------------------------------------------------------------

    static JNINativeMethod method_table[] = {
            { "initInputDevices_native", "()I", (void*)initInputDevices_native },
            { "initOutputDevices_native", "()I", (void*)initOutputDevices_native },
            { "getSwitchStatus_native", "(I)[Z", (void*)getSwitchStatus_native },
            { "writeDotMatrix_native", "(I[Z)V", (void*)writeDotMatrix_native },
            { "writeFND_native", "(II)V", (void*)writeFND_native },
            { "writeLCD_native", "(ILjava/lang/String;)V", (void*)writeLCD_native },
            { "writeLED_native", "(I[Z)V", (void*)writeLED_native },
            { "writeBUZZER_native", "(II)V", (void*)writeBUZZER_native },
            { "writeMOTOR_native", "(IIII)V", (void*)writeMOTOR_native },
            { "endInputDevices_native", "(I)V", (void*)endInputDevices_native },
            { "endOutputDevices_native", "(I)V", (void*)endOutputDevices_native }
    };

    int register_android_server_HuinsSystemController(JNIEnv *env){
        return jniRegisterNativeMethods(env, "com/android/server/HuinsSystemController", /* TODO java
            system service class path ex: com/android/server/LightsService */
                method_table, NELEM(method_table));
    }
};
```

system call: ioctl

```
.
└── frameworks
    └── base
        ├── Android.mk
        ├── core
        │   └── java
        │       └── android
        │           ├── app
        │           │   └── ContextImpl.java
        │           ├── content
        │           │   └── Context.java
        │           └── os
        │               ├── HuinsSystemManager.java
        │               └── IHuinsSystemService.aidl
        └── services
            ├── java
            │   └── com
            │       └── android
            │           └── server
            │               ├── huins
            │               │   ├── HuinsInputSender.java
            │               │   ├── HuinsSystemController.java
            │               │   └── HuinsSystemServiceImpl.java
            │               └── SystemServer.java
            └── jni
                ├── Android.mk
                ├── com_android_server_huins_HuinsSystemController.cpp
                ├── com_android_server_huins_HuinsSystemController.h
                └── onload.cpp
```

```
LOCAL_SRC_FILES:= \
    com_android_server_AlarmManagerService.cpp \
    com_android_server_AssetAtlasService.cpp \
    com_android_server_ConsumerIrService.cpp \
    com_android_server_input_InputApplicationHandle.cpp \
    com_android_server_input_InputManagerService.cpp \
    com_android_server_input_InputWindowHandle.cpp \
    com_android_server_LightsService.cpp \
    com_android_server_power_PowerManagerService.cpp \
    com_android_server_SerialService.cpp \
    com_android_server_SystemServer.cpp \
    com_android_server_UsbDeviceManager.cpp \
    com_android_server_UsbHostManager.cpp \
    com_android_server_VibratorService.cpp \
    com_android_server_location_GpsLocationProvider.cpp \
    com_android_server_location_FlpHardwareProvider.cpp \
    com_android_server_connectivity_Vpn.cpp \
    com_android_server_huins_HuinsSystemController.cpp \
    onload.cpp
# add Huins system service
```

```cpp
using namespace android;

extern "C" jint JNI_OnLoad(JavaVM* vm, void* reserved)
{
    JNIEnv* env = NULL;
    jint result = -1;

    if (vm->GetEnv((void**) &env, JNI_VERSION_1_4) != JNI_OK) {
        ALOGE("GetEnv failed!");
        return result;
    }
    ALOG_ASSERT(env, "Could not retrieve the env!");

    register_android_server_PowerManagerService(env);
    register_android_server_SerialService(env);
    register_android_server_InputApplicationHandle(env);
    register_android_server_InputWindowHandle(env);
    register_android_server_InputManager(env);
    register_android_server_LightsService(env);
    register_android_server_AlarmManagerService(env);
    register_android_server_UsbDeviceManager(env);
    register_android_server_UsbHostManager(env);
    register_android_server_VibratorService(env);
    register_android_server_SystemServer(env);
    register_android_server_location_GpsLocationProvider(env);
    register_android_server_location_FlpHardwareProvider(env);
    register_android_server_connectivity_Vpn(env);
    register_android_server_AssetAtlasService(env);
    register_android_server_ConsumerIrService(env);
    register_android_server_HuinsSystemController(env);


    return JNI_VERSION_1_4;
}
```

```java
        try {
            Slog.i(TAG, "IdleMaintenanceService");
            new IdleMaintenanceService(context, battery);
        } catch (Throwable e) {
            reportWtf("starting IdleMaintenanceService", e);
        }

        try {
            Slog.i(TAG, "Print Service");
            printManager = new PrintManagerService(context);
            ServiceManager.addService(Context.PRINT_SERVICE, printManager);
        } catch (Throwable e) {
            reportWtf("starting Print Service", e);
        }

        if (!disableNonCoreServices) {
            try {
                Slog.i(TAG, "Media Router Service");
                mediaRouter = new MediaRouterService(context);
                ServiceManager.addService(Context.MEDIA_ROUTER_SERVICE, mediaRouter);
            } catch (Throwable e) {
                reportWtf("starting MediaRouterService", e);
            }
        }

        /*
         * Add Huins System server
         */
        try {
            Slog.i(TAG, "Starting Huins Syetem Service");
            ServiceManager.addService(Context.HUINS_SYSTEM_SERVICE, new
                HuinsSystemServiceImpl(context));
            Slog.i(TAG, "Huins System Service Started");
        } catch (Throwable e) {
            Slog.e(TAG, "Failure starting Huins systerm service", e);
        }
    }

    // Before things start rolling, be sure we have decided whether
    // we are in safe mode.
    final boolean safeMode = wm.detectSafeMode();
    if (safeMode) {
        ActivityManagerService.self().enterSafeMode();
        // Post the safe mode state in the Zygote class
        Zygote.systemInSafeMode = true;
        // Disable the JIT for the system_server process
        VMRuntime.getRuntime().disableJitCompilation();
    } else {
        // Enable the JIT for the system_server process
        VMRuntime.getRuntime().startJitCompilation();
    }

    // It is now time to start up the app processes...
```

system server run huins system service register to service manager

```java
        registerService(CAMERA_SERVICE, new ServiceFetcher() {
            public Object createService(ContextImpl ctx) {
                return new CameraManager(ctx);
            }
        });

        registerService(PRINT_SERVICE, new ServiceFetcher() {
            public Object createService(ContextImpl ctx) {
                IBinder iBinder = ServiceManager.getService(Context.PRINT_SERVICE);
                IPrintManager service = IPrintManager.Stub.asInterface(iBinder);
                return new PrintManager(ctx.getOuterContext(), service, UserHandle.myUserId(),
                        UserHandle.getAppId(Process.myUid()));
            }});

        registerService(CONSUMER_IR_SERVICE, new ServiceFetcher() {
            public Object createService(ContextImpl ctx) {
                return new ConsumerIrManager(ctx);
            }});

        /* add Huins System service */
        registerService(HUINS_SYSTEM_SERVICE, new ServiceFetcher() {
            public Object createService(ContextImpl ctx) {
            return HuinsSystemManager.getHuinsSystemService();
            }});
        }

    static ContextImpl getImpl(Context context) {
        Context nextContext;
        while ((context instanceof ContextWrapper) &&
                (nextContext=((ContextWrapper)context).getBaseContext()) != null) {
            context = nextContext;
        }
        return (ContextImpl)context;
    }
```

register system service to Context

```
.
└── frameworks
    └── base
        ├── Android.mk
        ├── core
        │   └── java
        │       └── android
        │           ├── app
        │           │   └── ContextImpl.java
        │           ├── content
        │           │   └── Context.java
        │           └── os
        │               ├── HuinsSystemManager.java
        │               └── IHuinsSystemService.aidl
        └── services
            ├── java
            │   └── com
            │       └── android
            │           └── server
            │               ├── huins
            │               │   ├── HuinsInputSender.java
            │               │   ├── HuinsSystemController.java
            │               │   └── HuinsSystemServiceImpl.java
            │               └── SystemServer.java
            └── jni
                ├── Android.mk
                ├── com_android_server_huins_HuinsSystemController.cpp
                ├── com_android_server_huins_HuinsSystemController.h
                └── onload.cpp
```

4: register to context

3: client interface

2: system service

1 :native jni

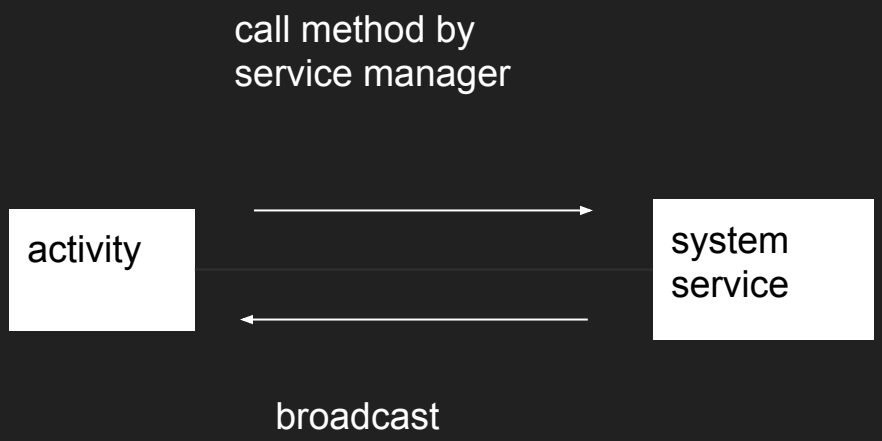
```java
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.IBinder;
import android.util.Log;
import android.os.IHuinsSystemService;


public class HuinsSystemManager {
    private final static String TAG = HuinsSystemManager.class.getName();
    private final static String SERVICE_NAME = Context.HUINS_SYSTEM_SERVICE;
    private final IHuinsSystemService service;
    private static HuinsSystemManager huinsManager;

    private boolean listeningState = false;

    public static synchronized HuinsSystemManager getHuinsSystemService() {
        if (huinsManager != null) {
            IBinder binder = android.os.ServiceManager.getService(SERVICE_NAME);
            if (binder != null) {
                IHuinsSystemService huinsService = IHuinsSystemService.Stub.asInterface(binder);
                huinsManager = new HuinsSystemManager(huinsService);
                try {
                    huinsService.init();
                } catch (android.os.RemoteException ex) {
                    Log.e(TAG, "Unable to initialize service");
                }
            }
        }
        return huinsManager;
    }

    private HuinsSystemManager(IHuinsSystemService service) {
        if(service == null){
            throw new IllegalArgumentException("Huins System service is null");
        }
        this.service = service;
    }

    public void startService() {
        try {
            if (!listeningState) service.startToListenSwitch();
        } catch (android.os.RemoteException ex) {
            Log.e(TAG, "service cannot start", ex);
        }
    }
```

get service from service manager

```java
package com.example.secureatm.mainMenu;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioButton;

import com.example.secureatm.R;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

import android.os.HuinsSystemManager;
import android.widget.TextView;


public class MainMenuActivity extends AppCompatActivity {

    private String LOG_TAG = MainMenuActivity.class.getName();
    private boolean[] ledPressed;
    private HuinsSystemManager huinsService;
    @BindView(R.id.FND_Input) public EditText FNDInput;
    @BindView(R.id.LCD_input) public EditText LCDInput;
    @BindView(R.id.SWITCH_OUTPUT) public TextView switch_output;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ButterKnife.bind( target: this);
        huinsService = (HuinsSystemManager) this.getApplicationContext()
                .getSystemService( s: "huins");

        HuinsSystemManager.HuinsInputReceiver receiver = huinsService.getHuinsInputReceiverInstance( context: this);
        receiver.setHandler(new HuinsSystemManager.HandlerSystemInput() {
            @Override
            public void handleInput(boolean[] booleans) {
                updateSwitchOutput(booleans);
            }
        });
        huinsService.startService();
    }
```

get system service

in client


call method by
service manager
activity
system
service
broadcast