Software Engineering Conference Russia
October 2017, St. Petersburg

Создание системных сервисов для платформы Android

Игорь Марков

Проект: собственный телефон

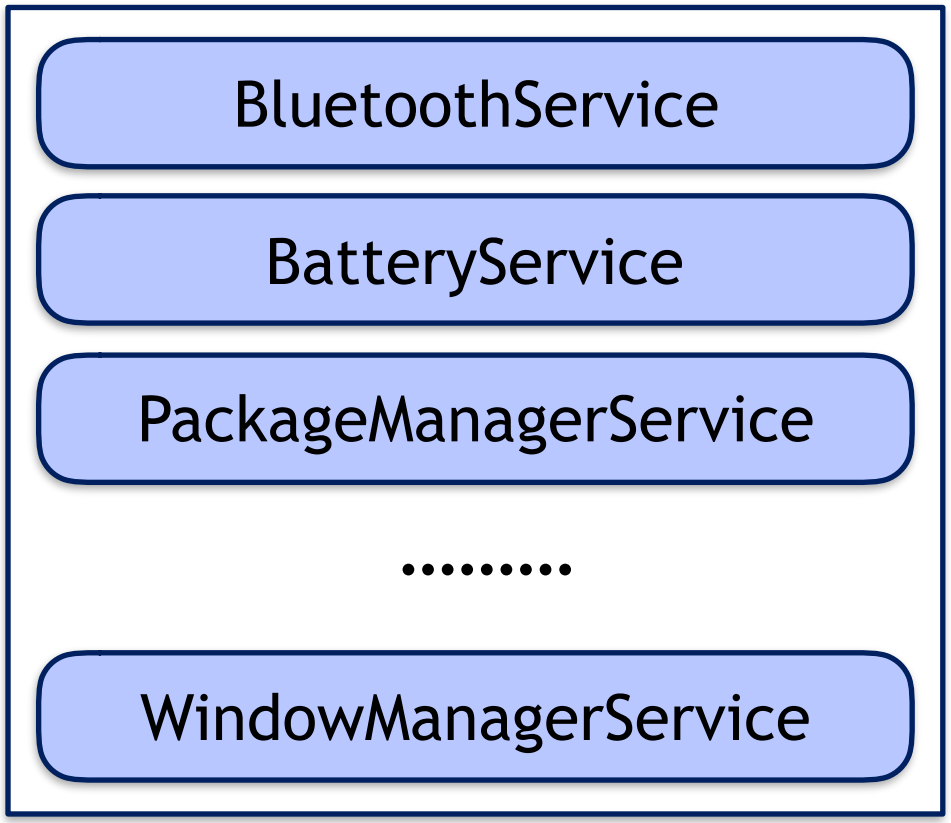
- Стартап
- Управление светодиодами
- Состояние боковых контактов
- Специализированное устройство тактильного отклика

Системные сервисы Android

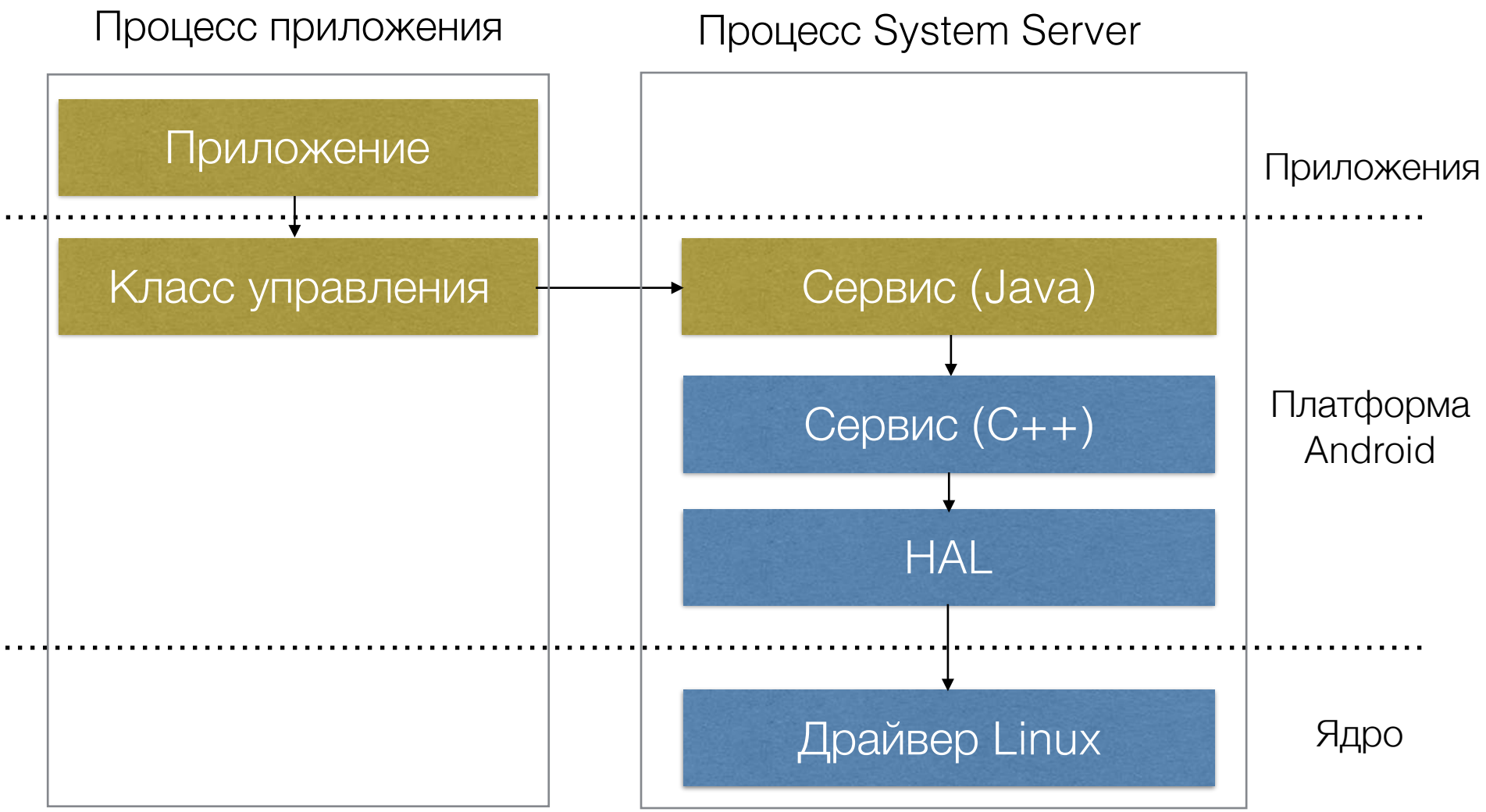
- Управляют аппаратным обеспечением
- Управляют ОС

Работают в процессе System Server

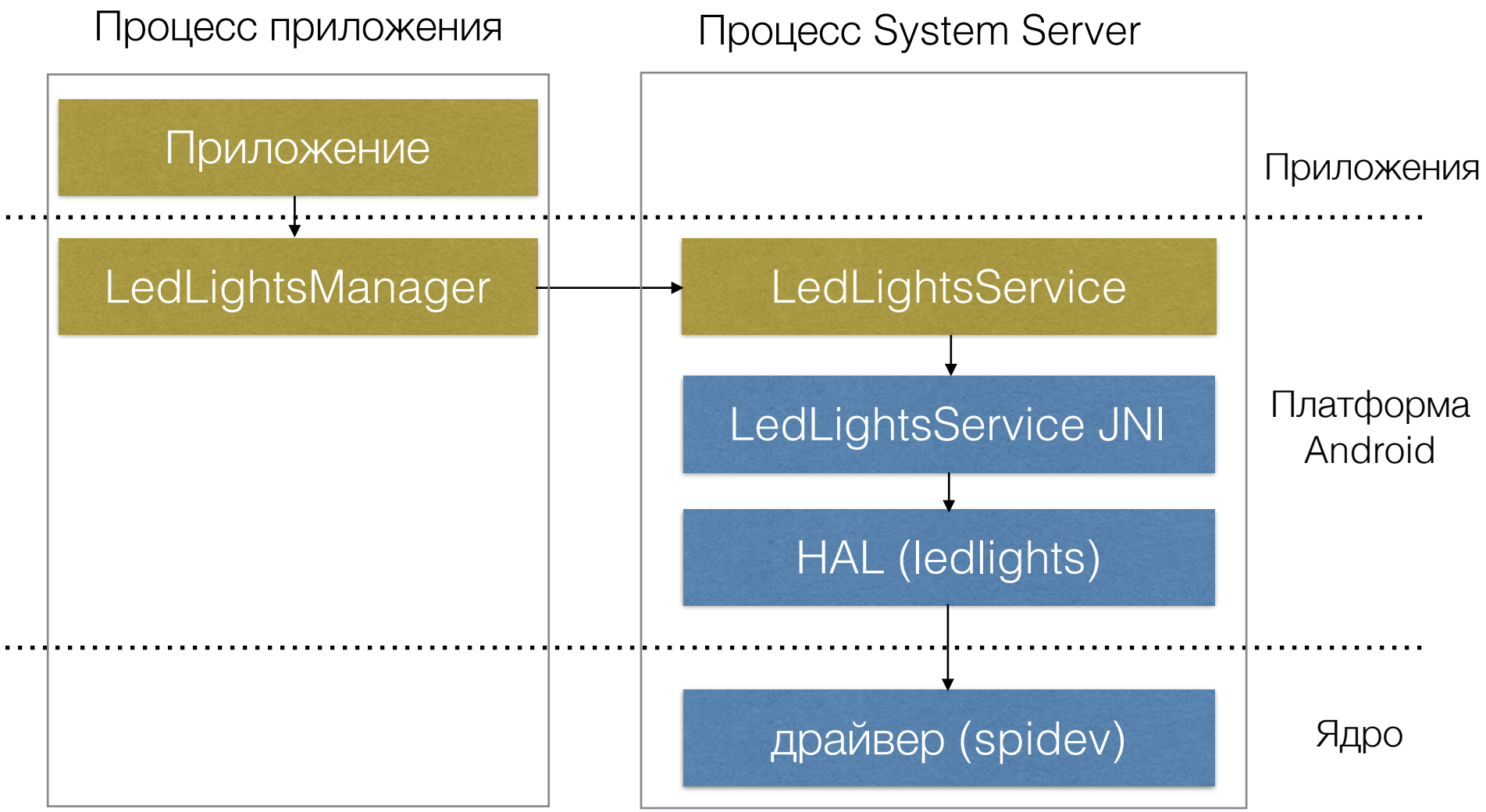
- Отдельный процесс Linux
- Больше привелегий



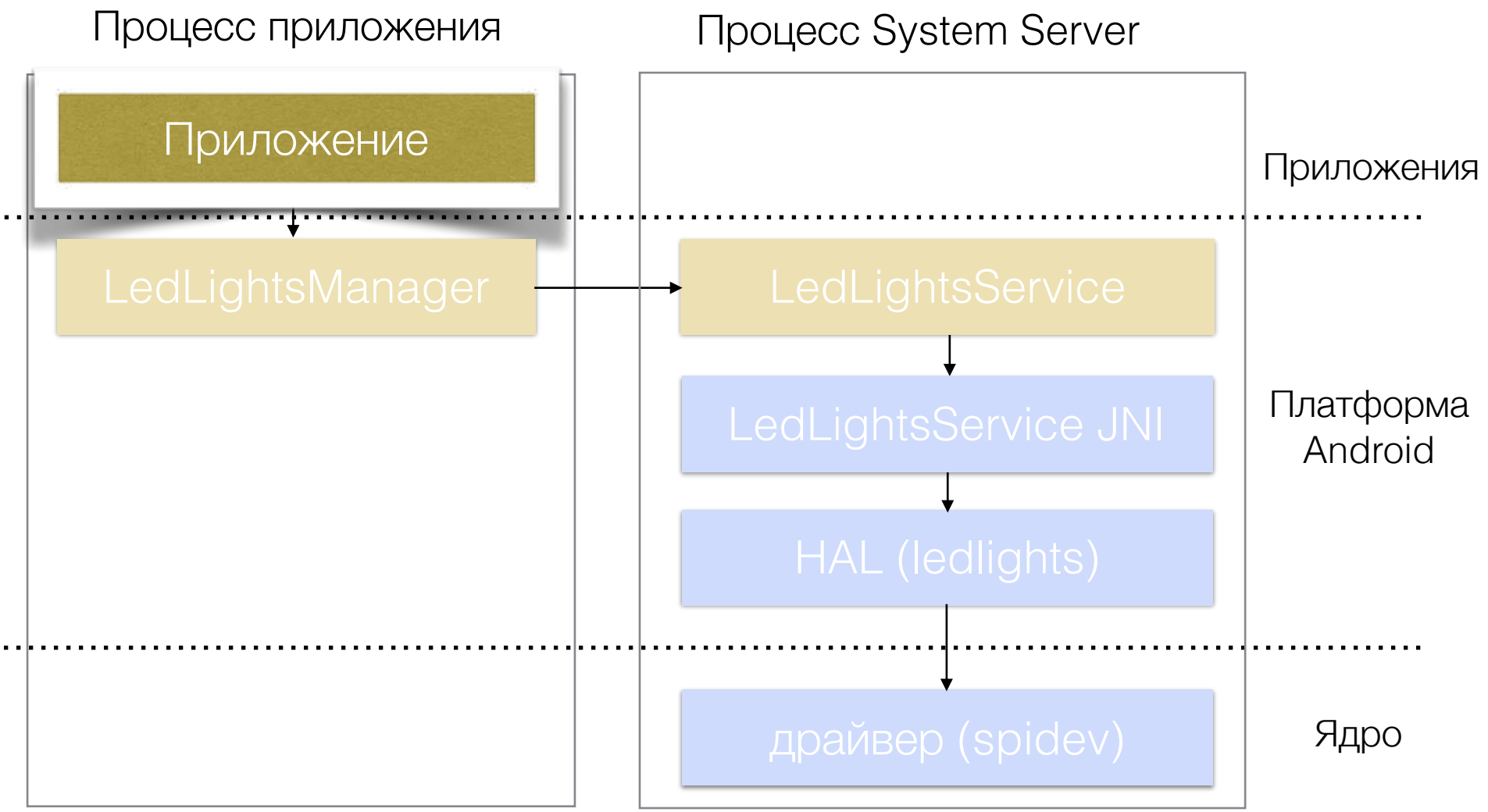
Общая схема


Процесс приложения
Процесс System Server
Приложение
Класс управления
Сервис (Java)
Сервис (C++)
HAL
Драйвер Linux
Приложения
Платформа Android
Ядро

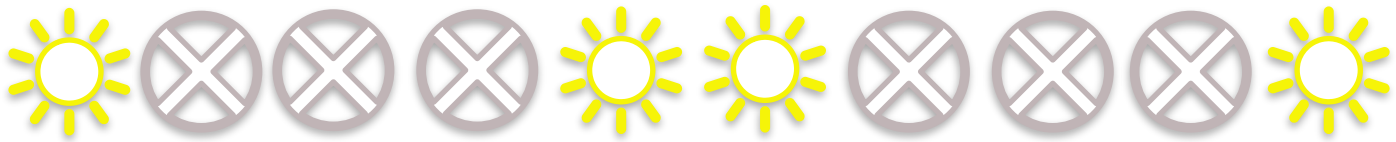
Сервис работы с светодиодами



Приложение

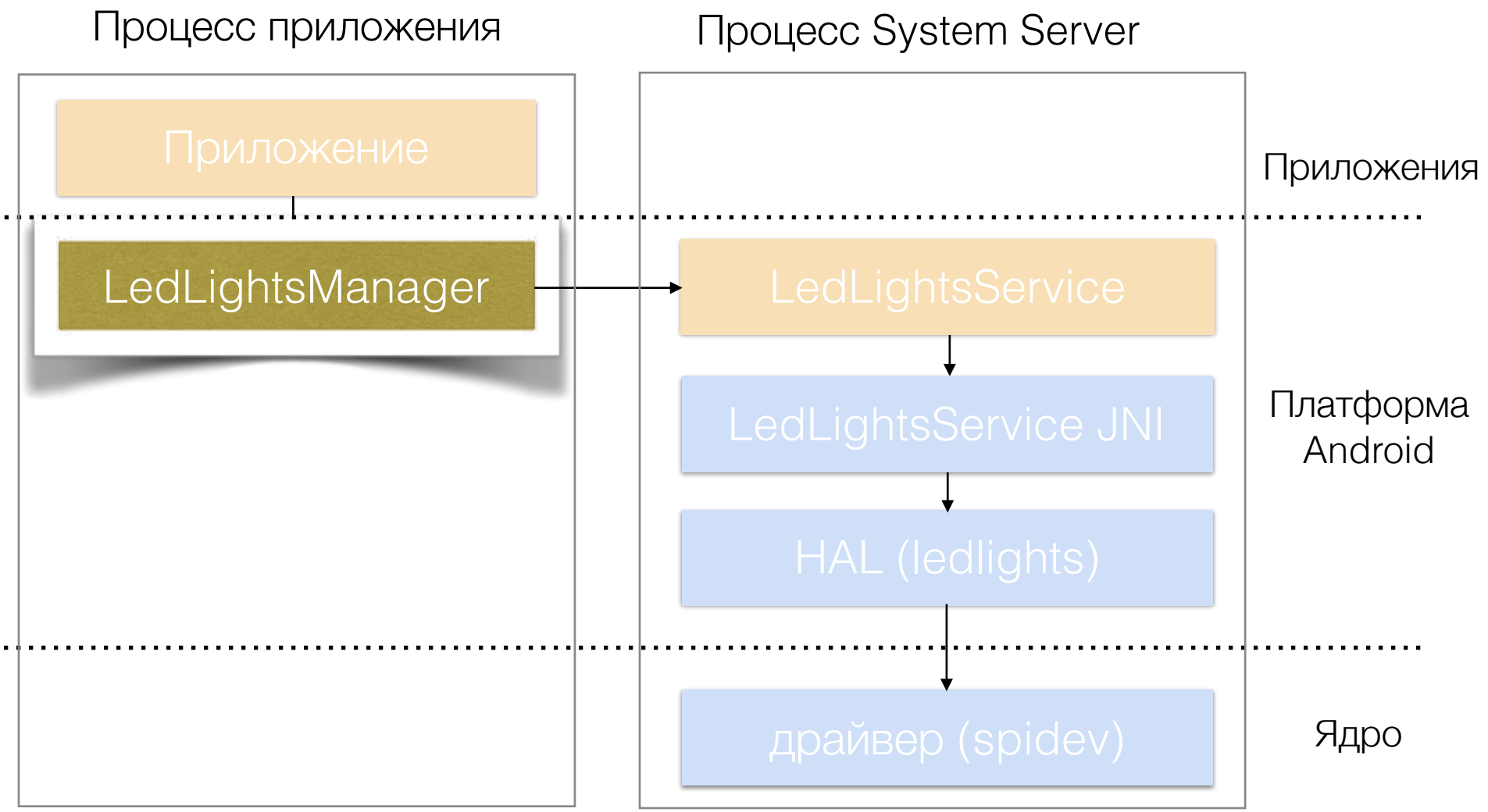

Процесс приложения
Процесс System Server
Приложение
LedLightsManager
LedLightsService
LedLightsService JNI
HAL (ledlights)
драйвер (spidev)
Приложения
Платформа Android
Ядро

Приложение



```
LedLightsManager manager =
      getSystemService(Context.LEDLIGHTS_SERVICE)
manager.setLights(new byte[]
    {100, 0, 0, 0, 100, 100, 0, 0, 0, 100});
```

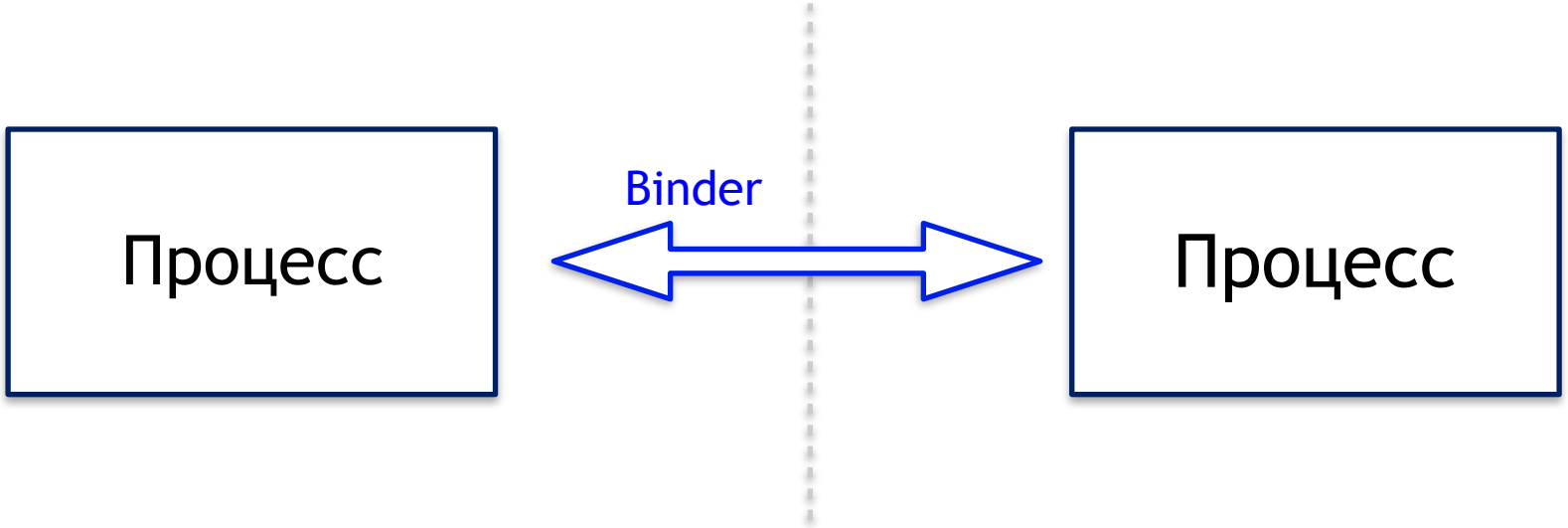
Управление сервисом



Класс управления сервисом

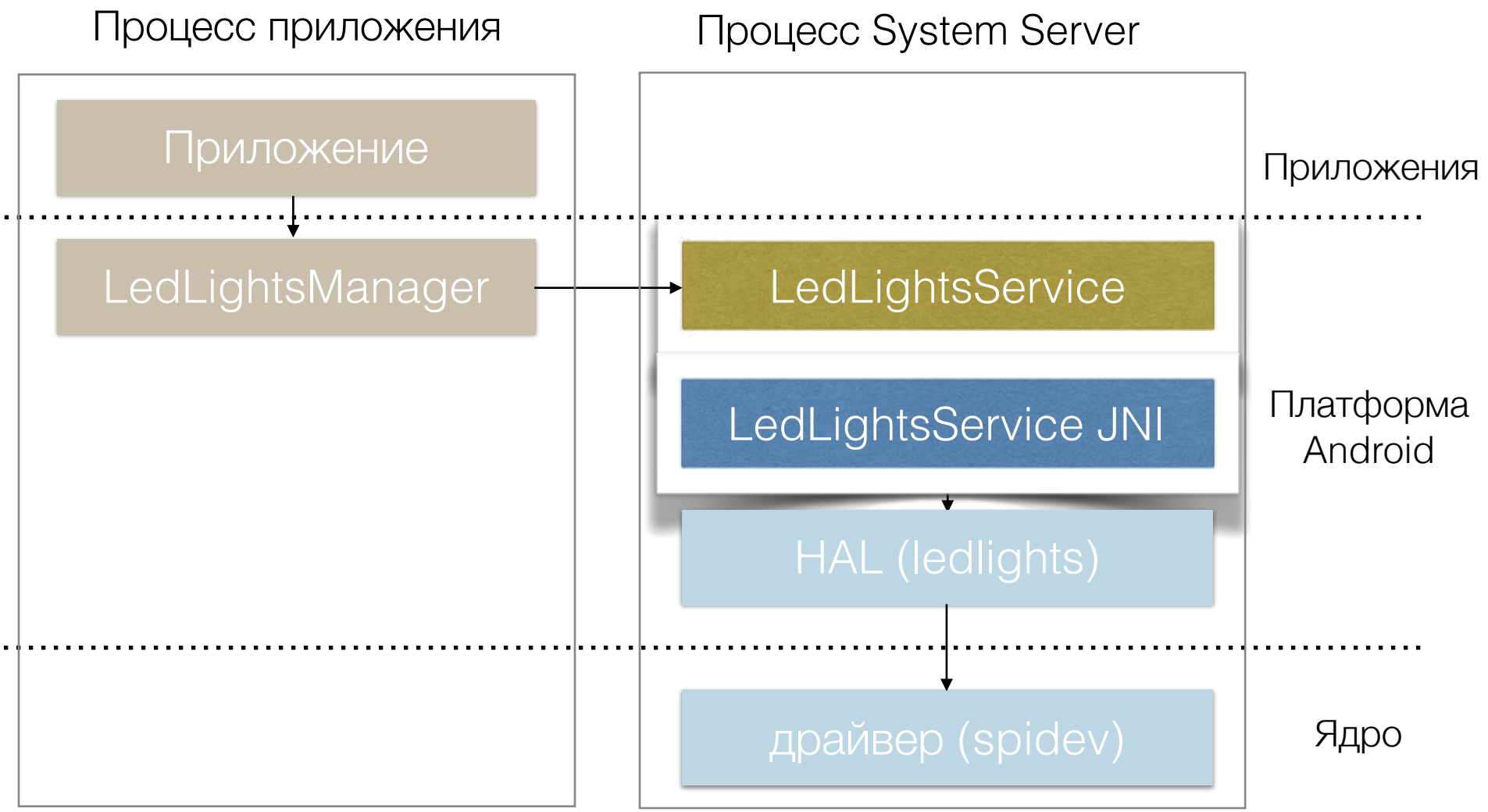
```
public class LedLightsManager {

  public static LedLightsManager getInstance() {
     …
     serviceInstance = ILedLightsService.Stub.asInterface(
        ServiceManager.getService(Context.LEDLIGHTS_SERVICE));
     …
  }

  public void setLights(byte[] leds) {
    try {
      serviceInstance.setLights(leds);
    } catch (RemoteException e) {}
  }

}
```

Binder, AIDL



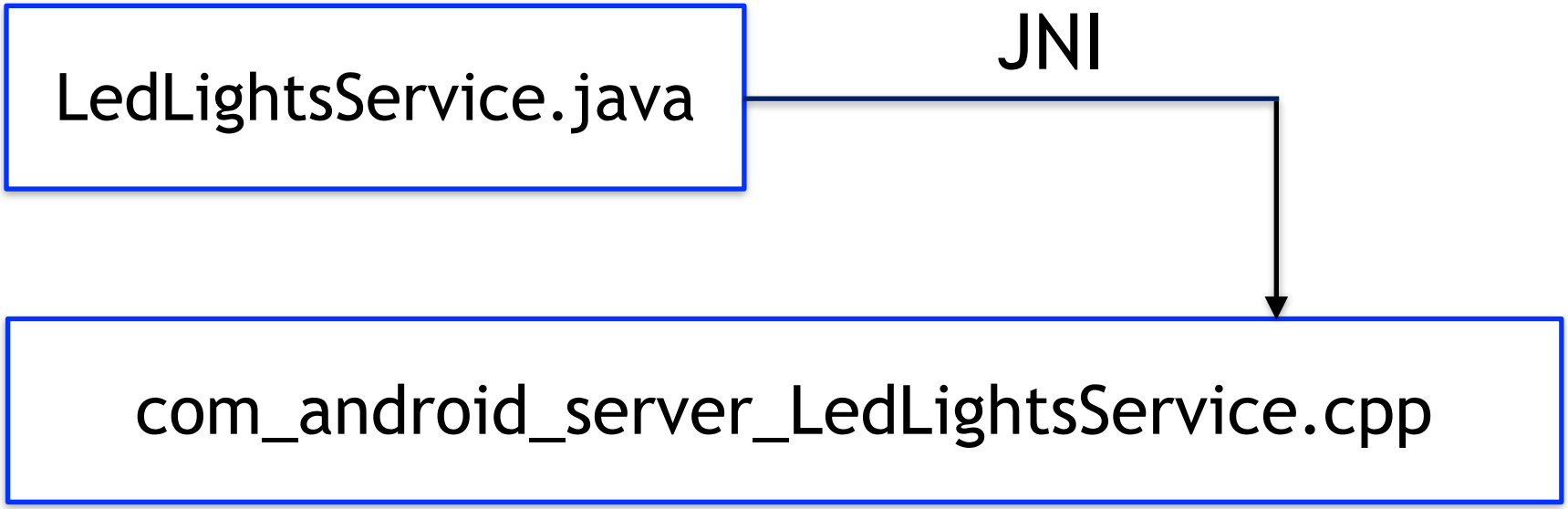

```
package android.hardware;
/** @hide */
interface ILedLightsService
{
        oneway void setLights(in byte[] leds);
}
```

Системный сервис



Системный сервис

- Часть на Java для логики и интерфейса к приложению
- Часть на C/C++ для доступа к модулю HAL



Код сервиса (Java)

```
public class LedLightsService extends
                              ILedLightsService.Stub {

  public void setLights(byte[] leds) {
    if (leds.length == 10)
      nativeSetLeds(leds);
  }

  private native int nativeSetLeds(byte[] leds);
}
```

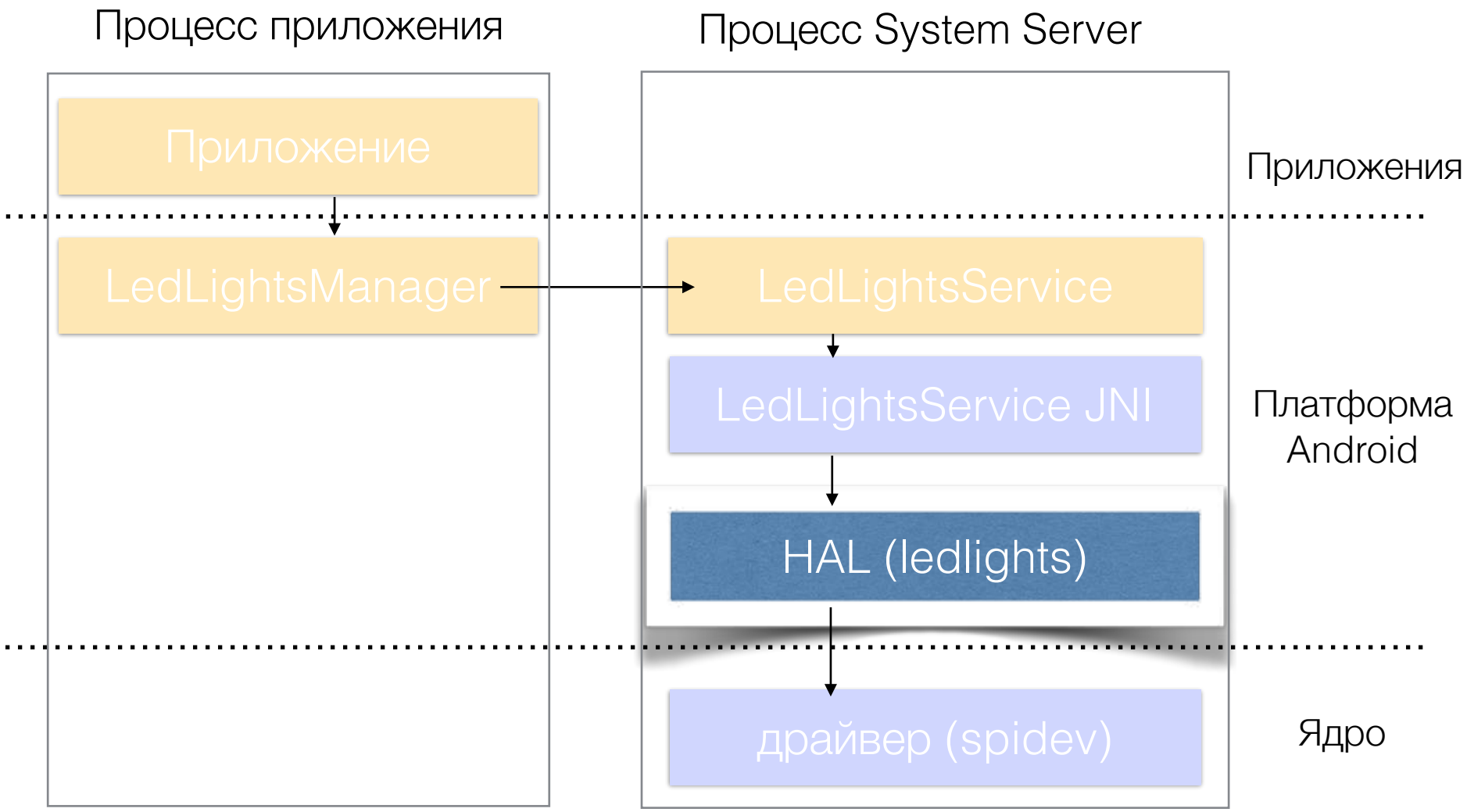
Инициализация HAL из сервиса

```
static jint initHAL(JNIEnv *env, object clazz)
{

  hw_module *module;
  int res = hw_get_module(LEDLIGHTS_HARDWARE_MODULE_ID,
                          (const hw_module_t**)&module);
  if (res == 0)
    res = module->methods->open(module,
                                LEDLIGHTS_HARDWARE_MODULE_ID,
                                (hw_device_t**)&device);
  return res;

}
```

Отправка данных в HAL

```
static jint nativeSetLeds(JNIEnv *env, jobject jobj,
                                       jbyteArray jarr)
{
  int res = -1;

  jbyte *buf = env->GetByteArrayElements(jarr, NULL);
  if (buf) {
    res = device->write((char *)buf);
    env->ReleaseByteArrayElements(jarr, buf, JNI_ABORT);
  }
  return res;

}
```

Hardware Abstraction Layer



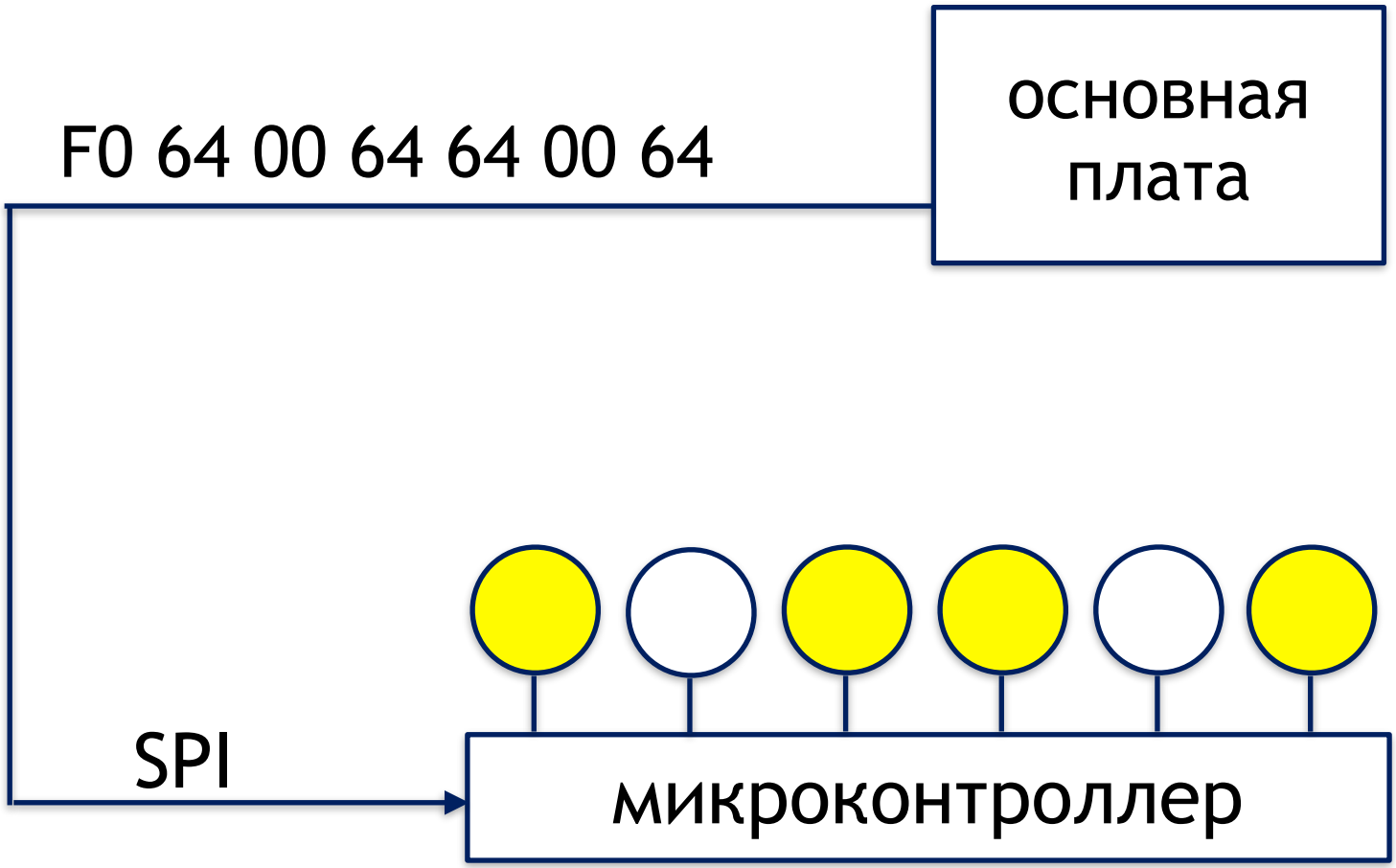
Модуль HAL

- Является «драйвером»
- В отличие от Linux, выполняется в userspace
- Общий интерфейс для одного типа устройств

HAL, функция записи

```
static int liblights_write(char *buffer)
{
    char cmd[LIGHTS_NUM + 1];
    cmd[0] = 0xF0;
    memcpy(cmd + 1, buffer, LIGHTS_NUM);

    return ioctl(spi_fd, SPI_IOC_MESSAGE(1), &cmd);
}
```

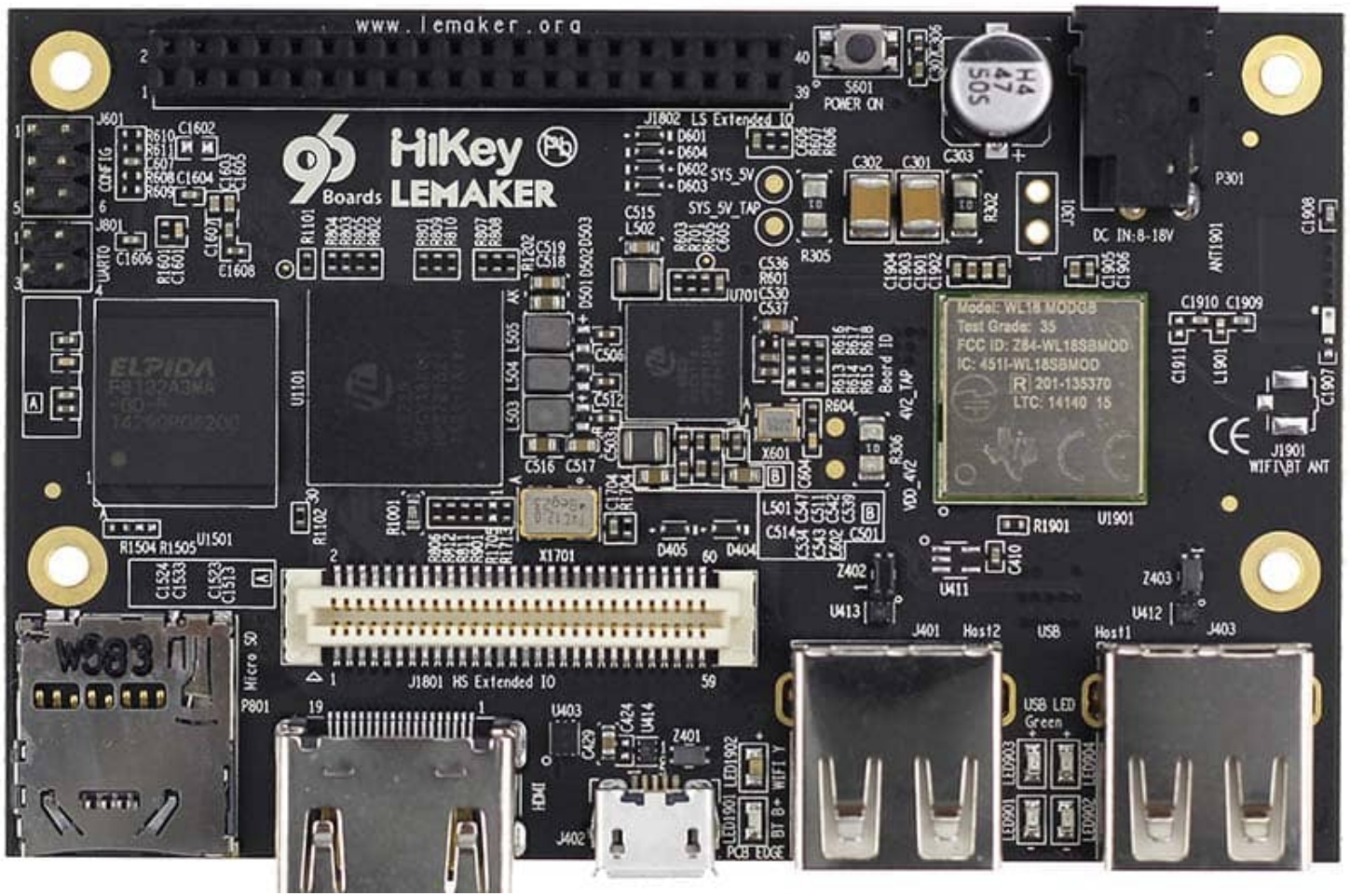
Аппаратное обеспечение



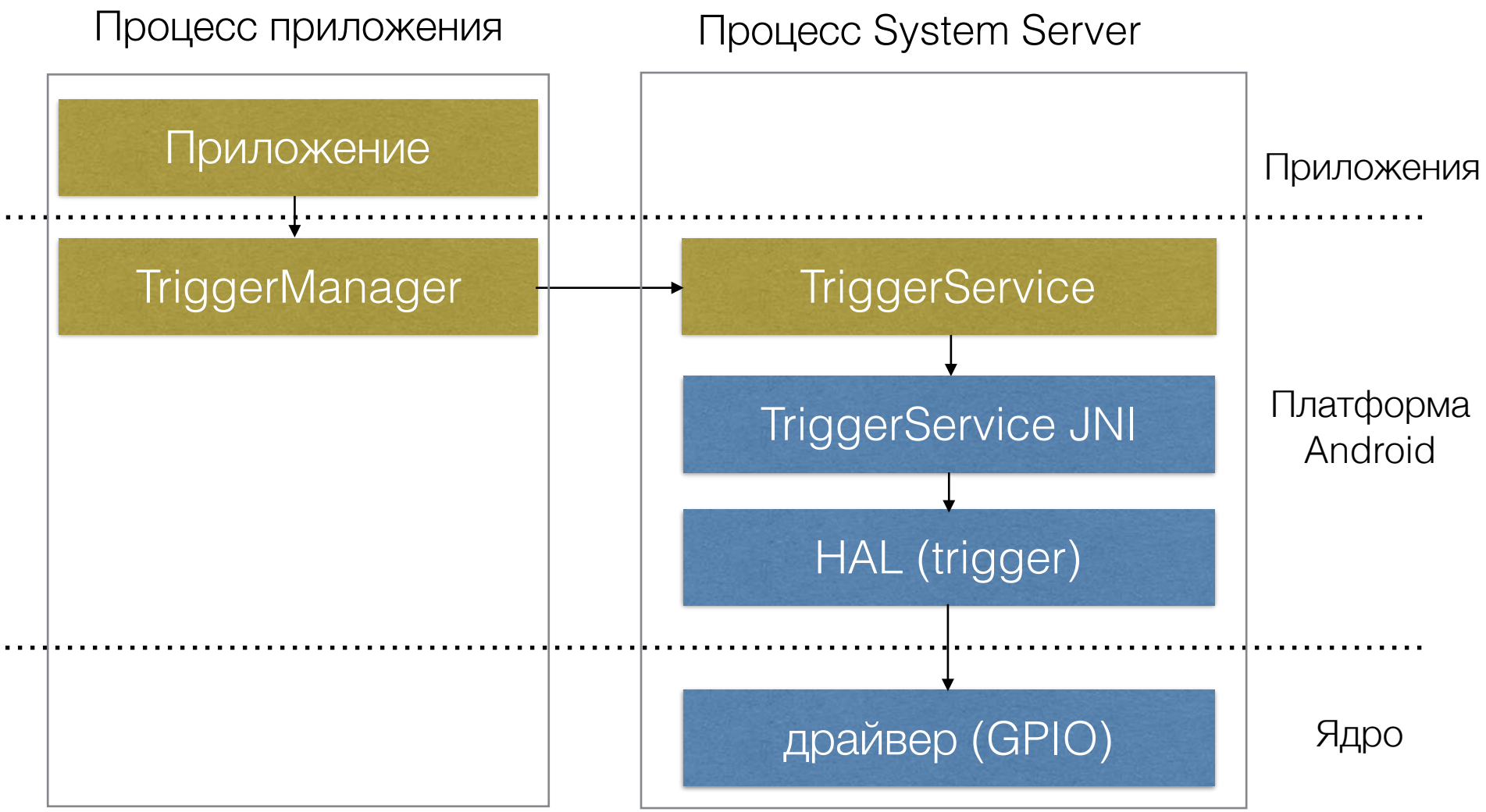
Сборка и прошивка

```
$ . build/evnsetup
$ lunch
$ make -j40 update-api
$ make -j40


$ adb reboot bootloader
$ fastboot flash system \
      out/target/product/YP/system.img
$ fastboot reboot
```

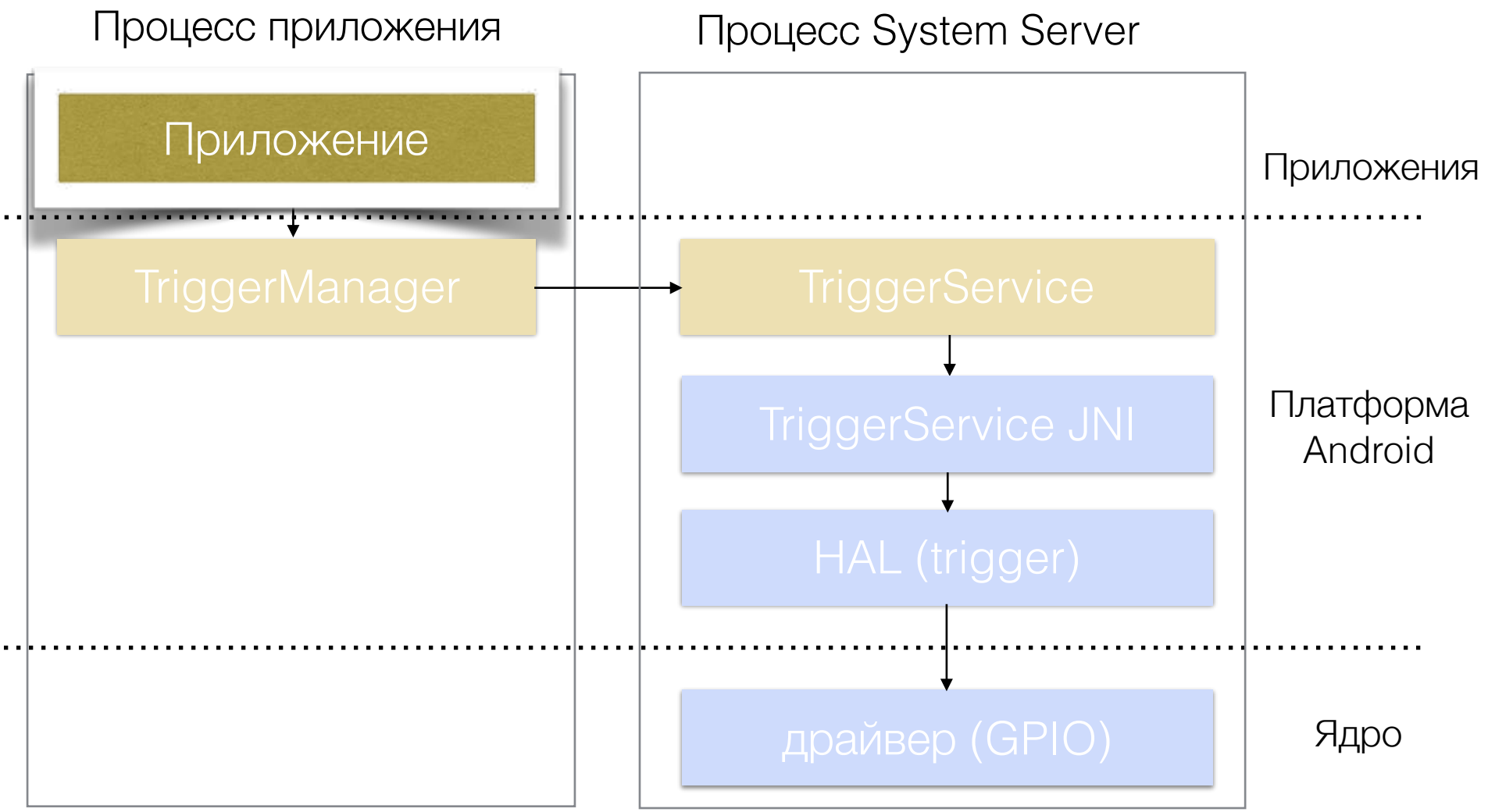
Можно в домашних условиях!



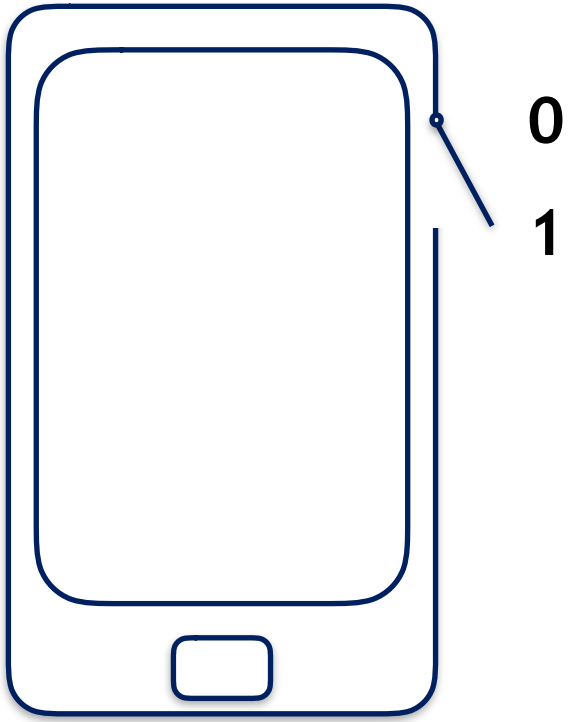
Сервис работы с контактами



Приложение

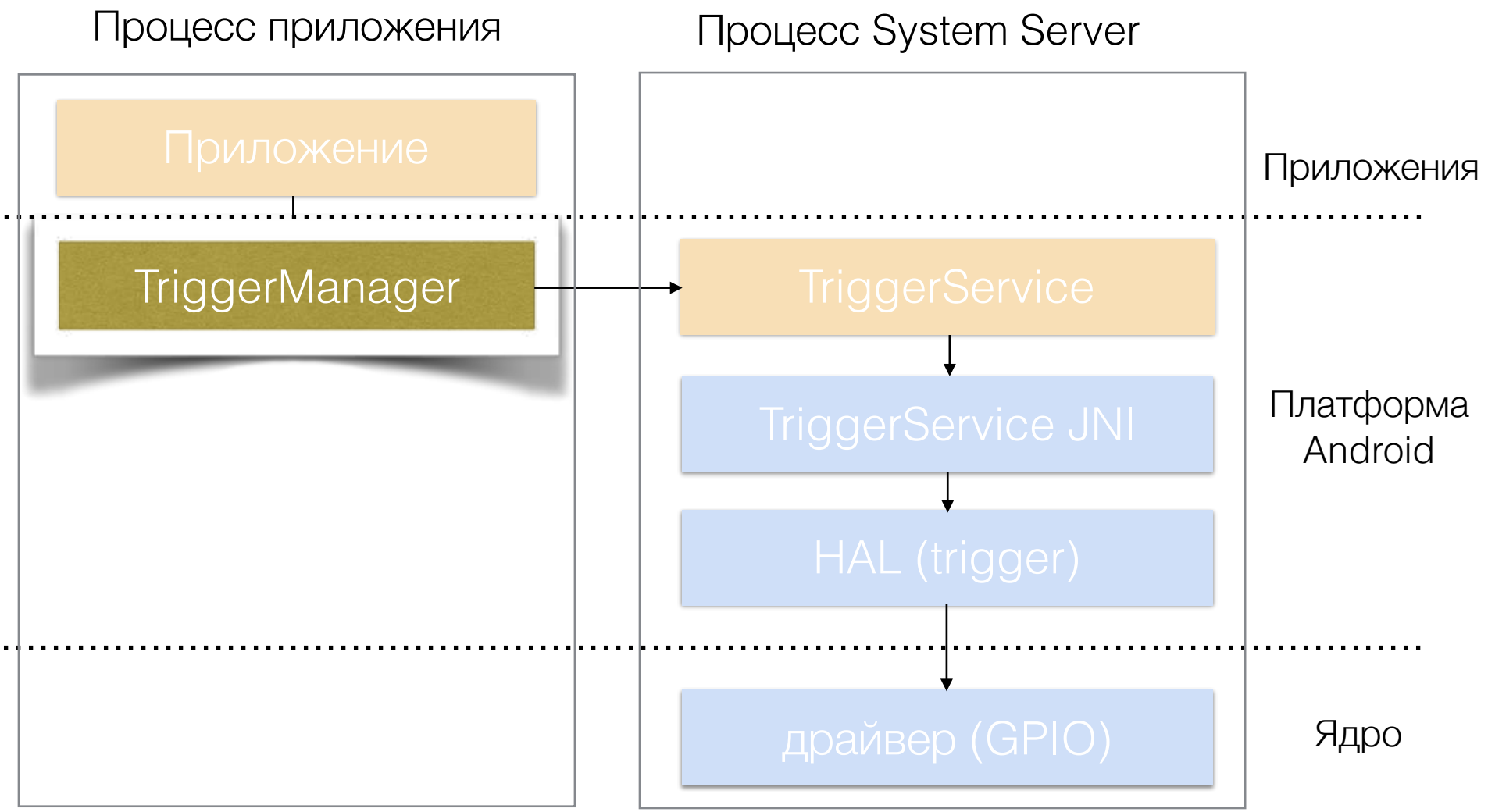

Процесс приложения
Процесс System Server
Приложение
TriggerManager
TriggerService
TriggerService JNI
HAL (trigger)
драйвер (GPIO)
Приложения
Платформа Android
Ядро

Приложение




```
TriggerManager manager =
      getSystemService(Context.TRIGGER_SERVICE);
int state = manager.getState();
```

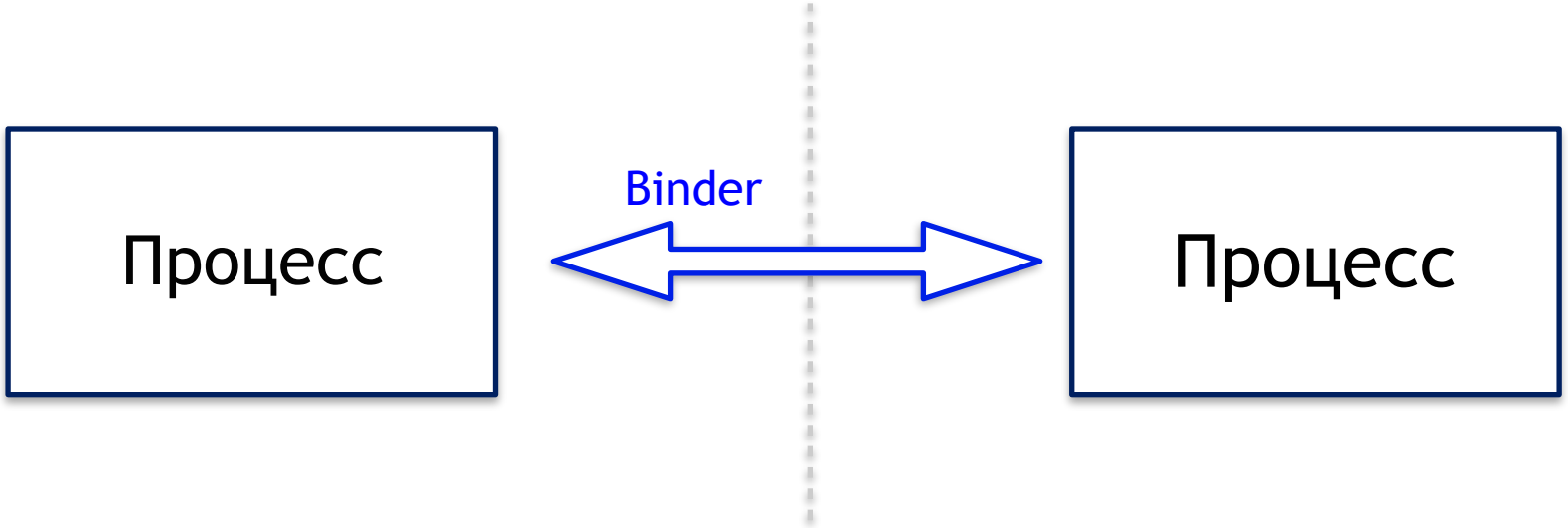
Управление сервисом



Класс управления сервисом

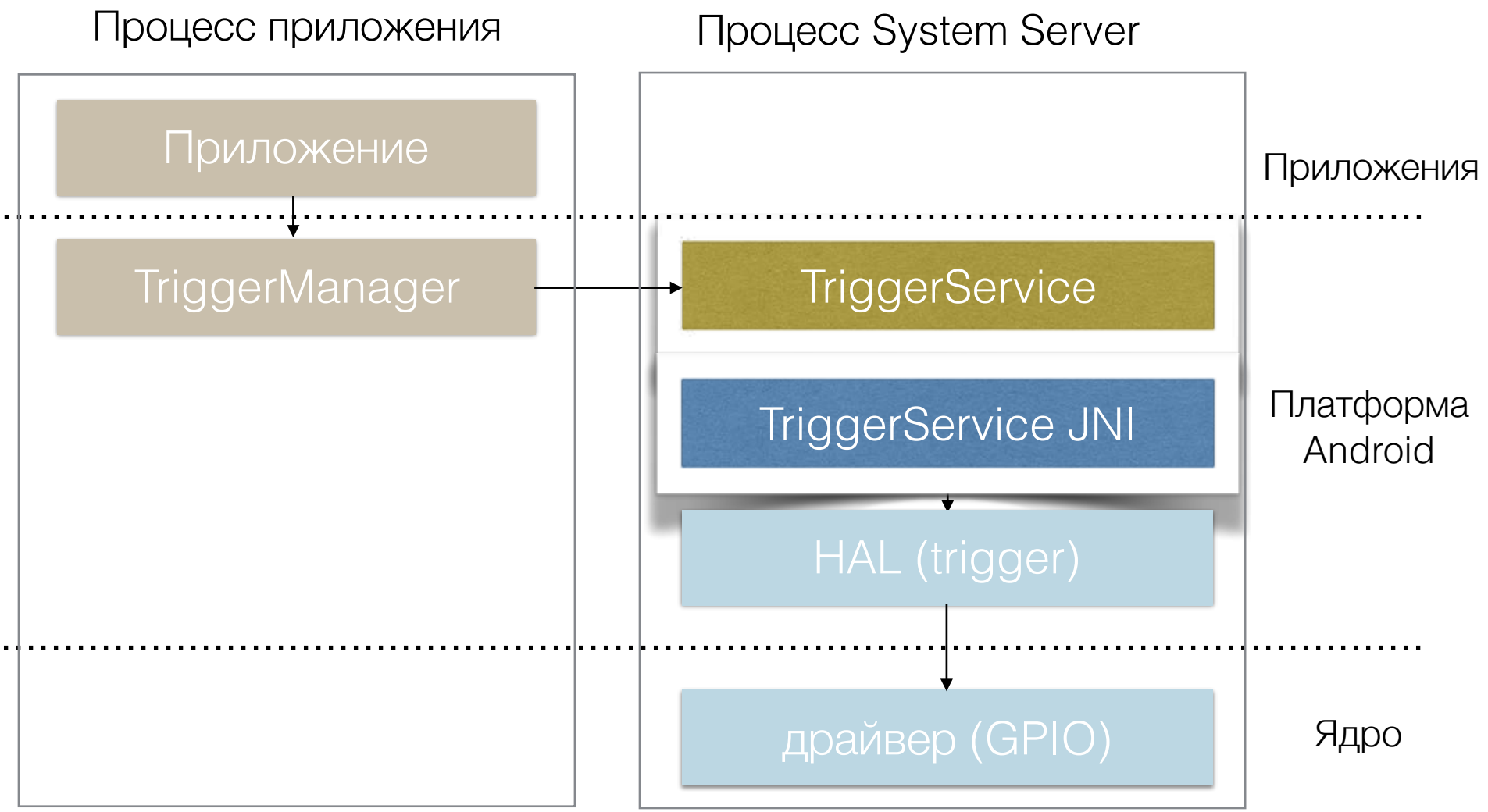
```
public class TriggerManager {

  public static TriggerManager getInstance() {
     …
     serviceInstance = ITriggerService.Stub.asInterface(
        ServiceManager.getService(Context.TRIGGER_SERVICE));
     …
  }

  public int getState() {
    try {
      return serviceInstance.getState();
    } catch (RemoteException e) {return 0;}
  }

}
```

Интрефейс AIDL




```
package android.hardware;
/** @hide */
interface ITriggerService {

        int getState();

}
```

Системный сервис



Код сервиса (Java)

```
public class TriggerService extends
                                    ITriggerService.Stub {

  public int getState() {
       return nativeGetState();
  }

  private native int nativeGetState();
}
```

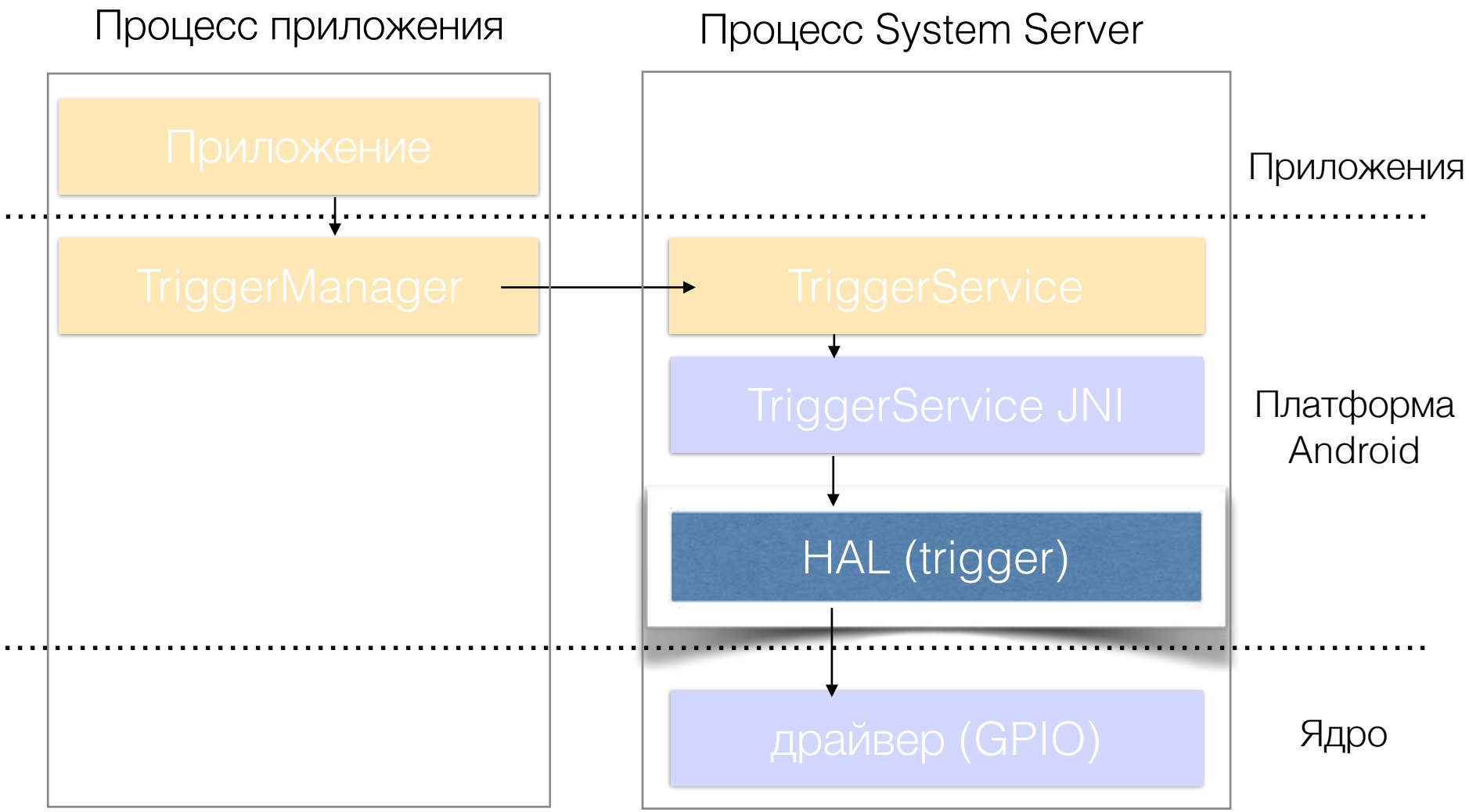
Инициализация HAL из сервиса

```
static jint initHAL(JNIEnv *env, object clazz)
{

  hw_module *module;
  int res = hw_get_module(TRIGGER_HARDWARE_MODULE_ID,
                          (const hw_module_t**)&module);
  if (res == 0)
    res = module->methods->open(module,
                                TRIGGER_HARDWARE_MODULE_ID,
                                (hw_device_t**)&device);
  return res;

}
```

Отправка данных в HAL

```
static jint nativeGetState(JNIEnv *env, jobject jobj)
{
    return device->readState();
}
```

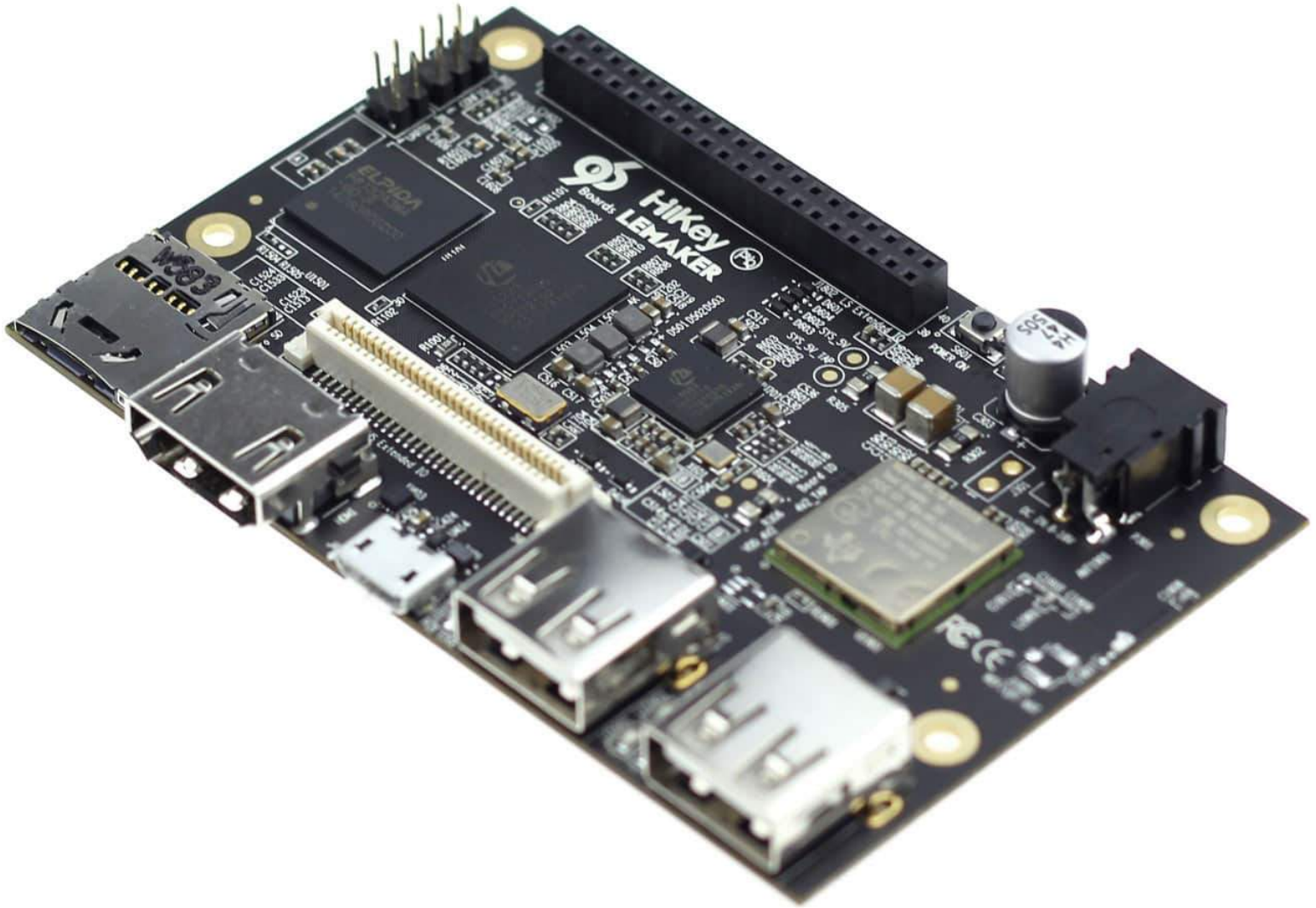
Hardware Abstraction Layer



HAL, функция чтения

```
static int trigger_readState()
{
   char buf[8];
   lseek(gpio_fd, 0, SEEK_SET);

   if (read(gpio_fd, &buf, sizeof(buf)) > 0
     return strcmp(buf, "1") == 0;
   else
     return -1;
}
```

Добавить системный сервис — это легко!



Если возникли любые вопросы
при реализации
системного сервиса, пишите

Игорь Марков
igor.markov@auriga.com

Эта презентация: http://bit.ly/2yzjhqt