Sensor Plug & Play
지원과제 성과발표

발 표 자 : 은 성 배

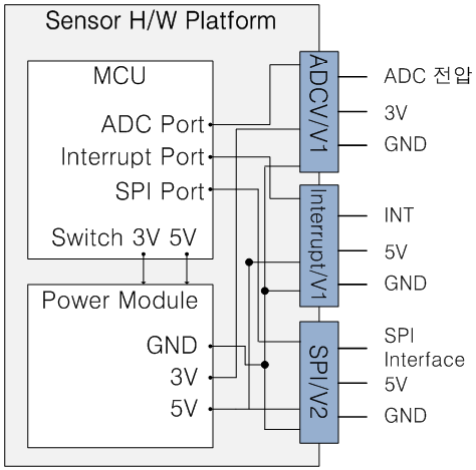
HCI Lab.
Hannam University

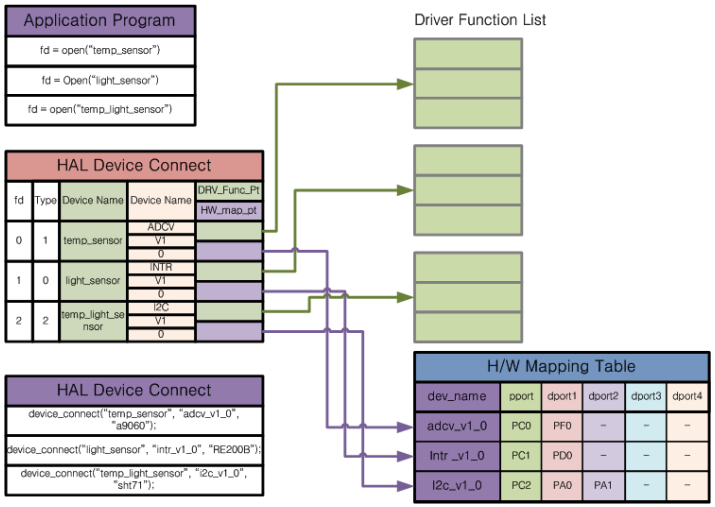
연 구 목 표

- 개방형 Plug & Play 개념의 로컬 인식형 센서 인터페이스를 위한 기술 연구
  - 로컬에서 인식하는 센서 인터페이스 구현
  - 센서 인터페이스에 대한 모듈 설계
  - 센서 식별 정보 전송에 관련한 프로토콜 구현
- 일반 펌웨어 상에서 위 구조를 구현
  - 일반 펌웨어 상에 센서 개방형 Plug & Play 구현
  - 센서 Plug & Play 실 수행 예제 구현
- 공개 소스 커뮤니티 활동 전개

센서 지원에 대한 연구

- 센서투명성을 지원하는 센서 노드용 운영체제 구조 [1]
- 센서 투명성을 지원하는 센서 디바이스 매니저 [2]
  - H/W ➔ 인터페이스, 전원 제어 지원
  - S/W ➔ 센서 디바이스 드라이버 지원
  - 사용자 ➔ 인터페이스 지정, 드라이버 지정





- [1] 은성배, 소신섭, 김병호 / 센서투명성을 지원하는 센서 노드 운영체제 구조 / 한국정보과학회 2008 종합학술대회 논문집 제 35권 제1호(A)
- [2]방상호, 은성배 / 센서 투명성을 지원하는 센서 디바이스 매니저 / 한국정보처리학회 2008 추계학술발표대회 논문집 제15권 제2호

센서 Plug & Play

시스템 설치 및 구성

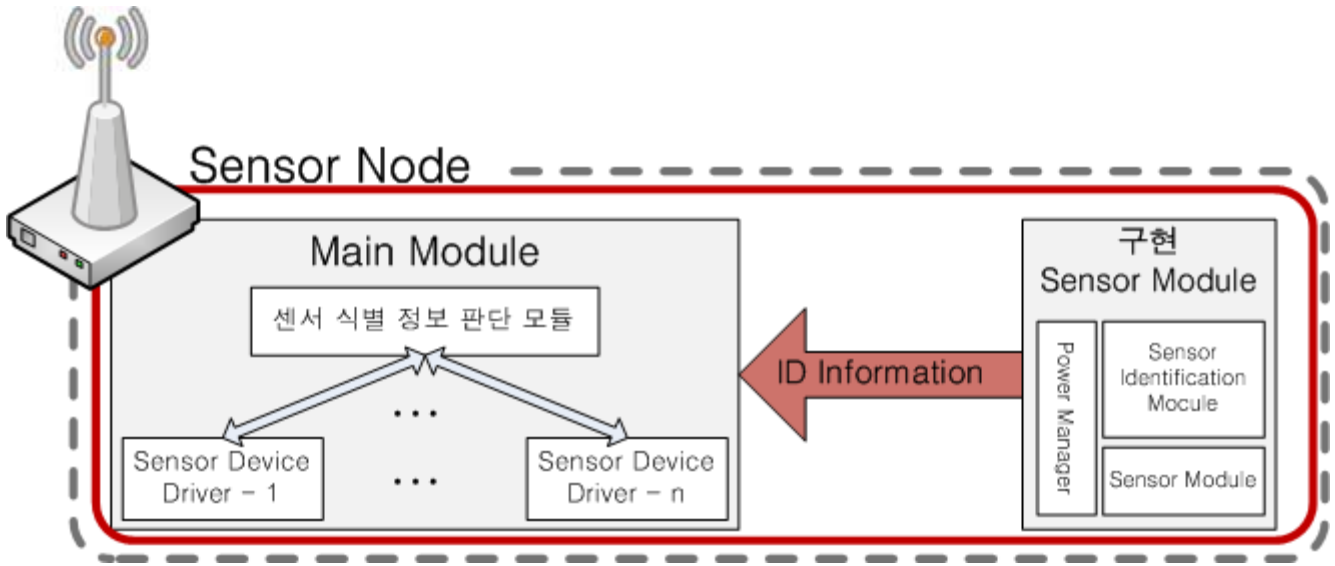
- 설치 단계는 전체 어플리케이션 개발 비용의 23% 소비[3]

센서 Plug & Play 장점

- 응용 개발자의 부담 덜어줌
- 빠른 시스템 설치
- 진단 기능 향상
- 센서 수리 및 교체로 인한 다운타임 감소

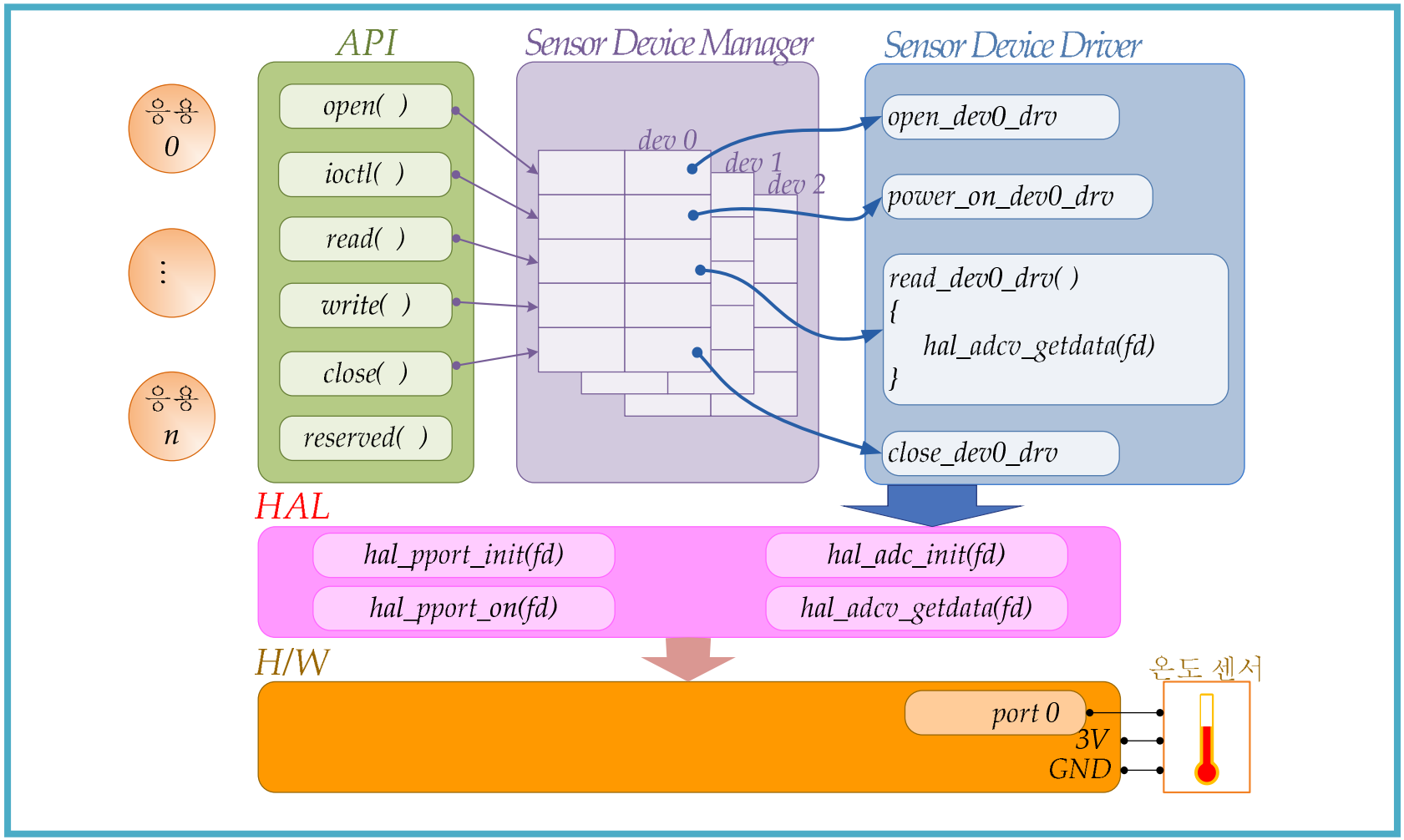
센서 Plug & Play

- Main Module ➔ Sensor Device Driver 보유
- 구현 Sensor Module ➔ Sensor Identification Information 보유



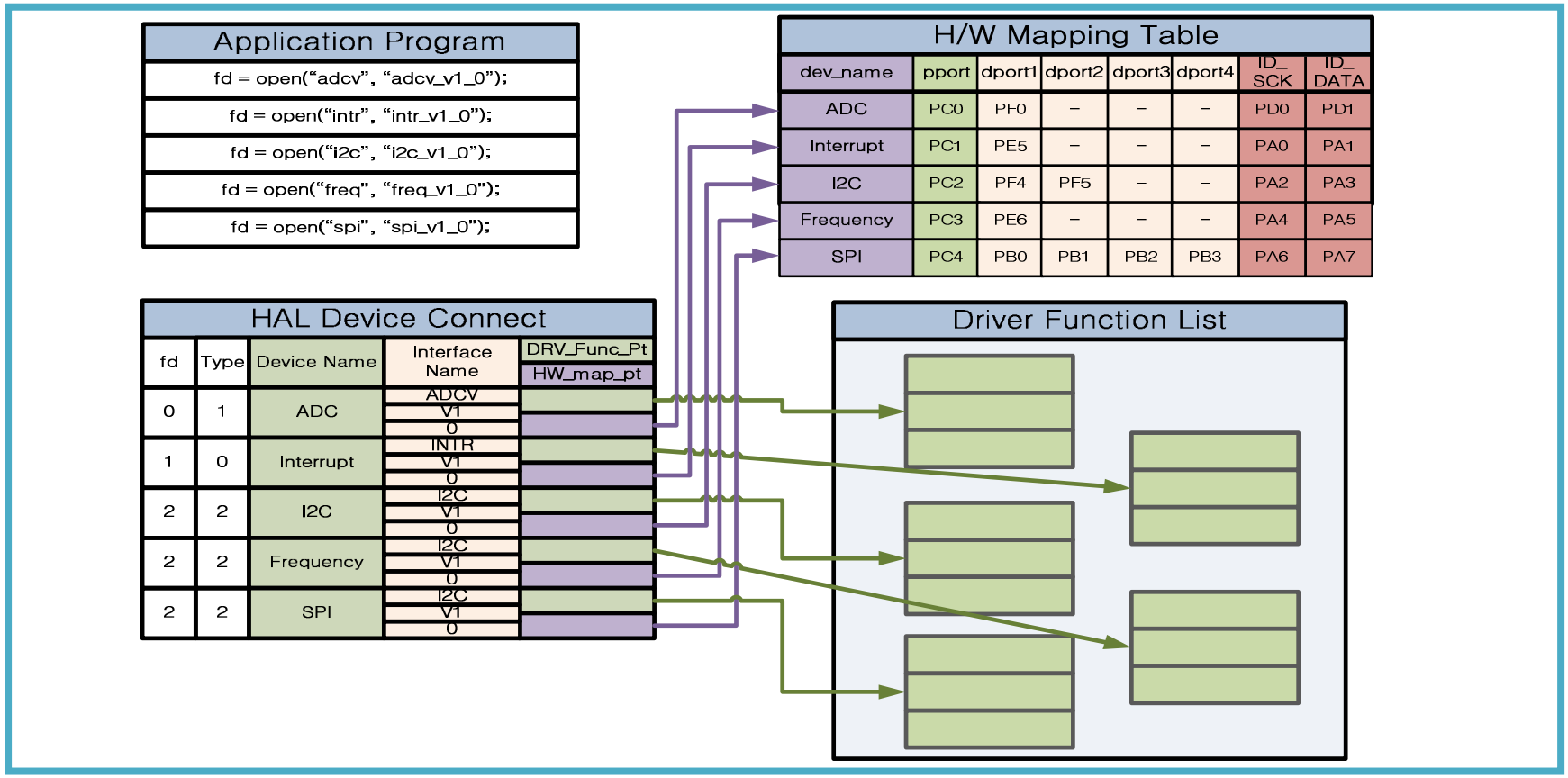
Plug & Play 설계 [1]

센서 노드 플랫폼 기반 구조



Plug & Play 설계 [2]

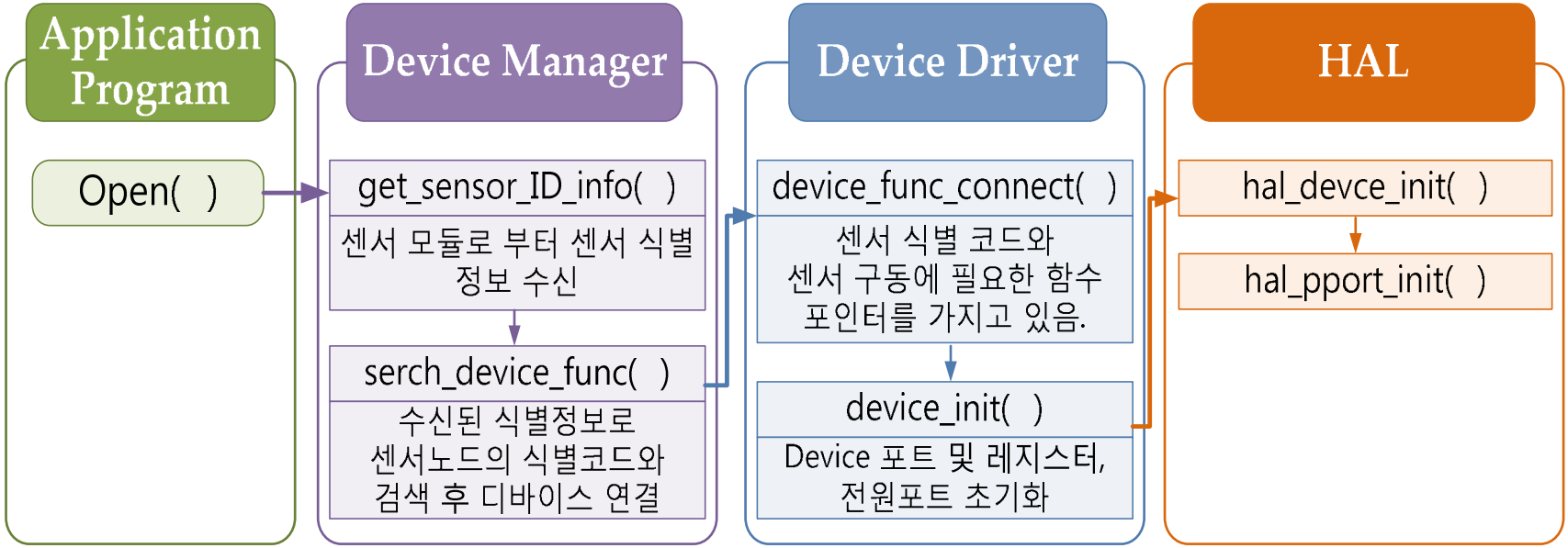
센서 D/D 매니저 구조



- Driver Function List, HAL Device Connect 및 H/W Mapping Table의 일부를 포함하고 있음.
- HAL Device Connect는 센서 디바이스 및 DD정의를 함.

Plug & Play 설계 [3]

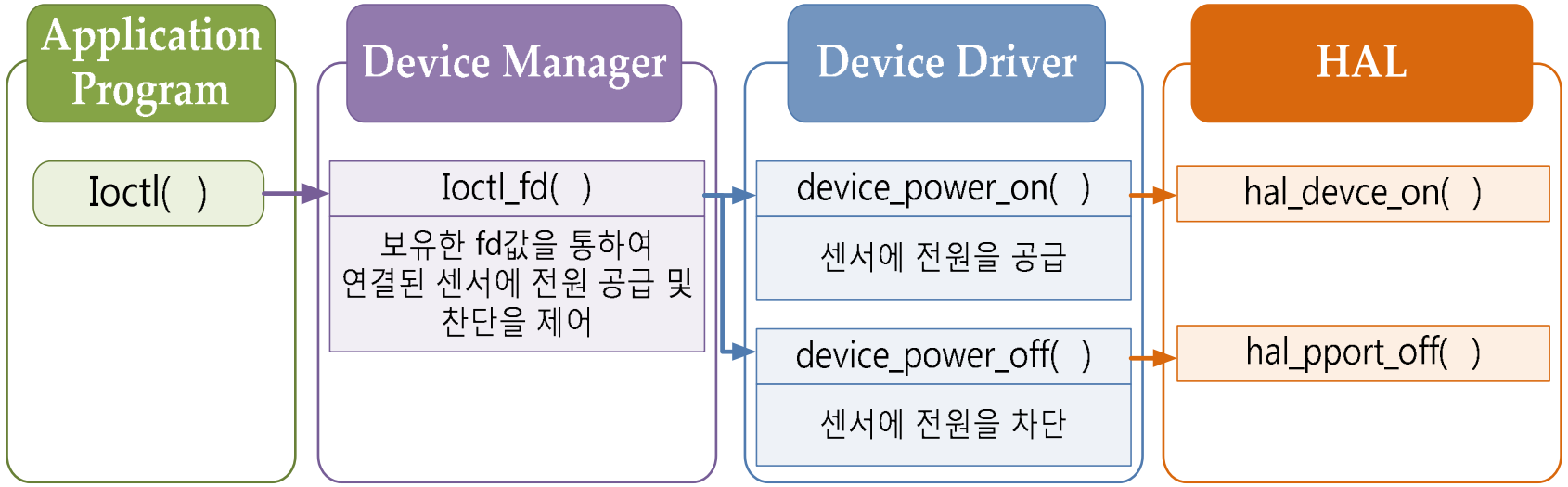
센서 디바이스 연결 과정



- Open( ) 함수를 통하여 센서모듈로 식별정보 수신 및 디바이스 연결
- Device Manager는 센서와 디바이스 드라이버를 연결 하는 역할 담당
- Device Driver는 HAL영역의 함수 호출을 담당

Plug & Play 설계 [4]

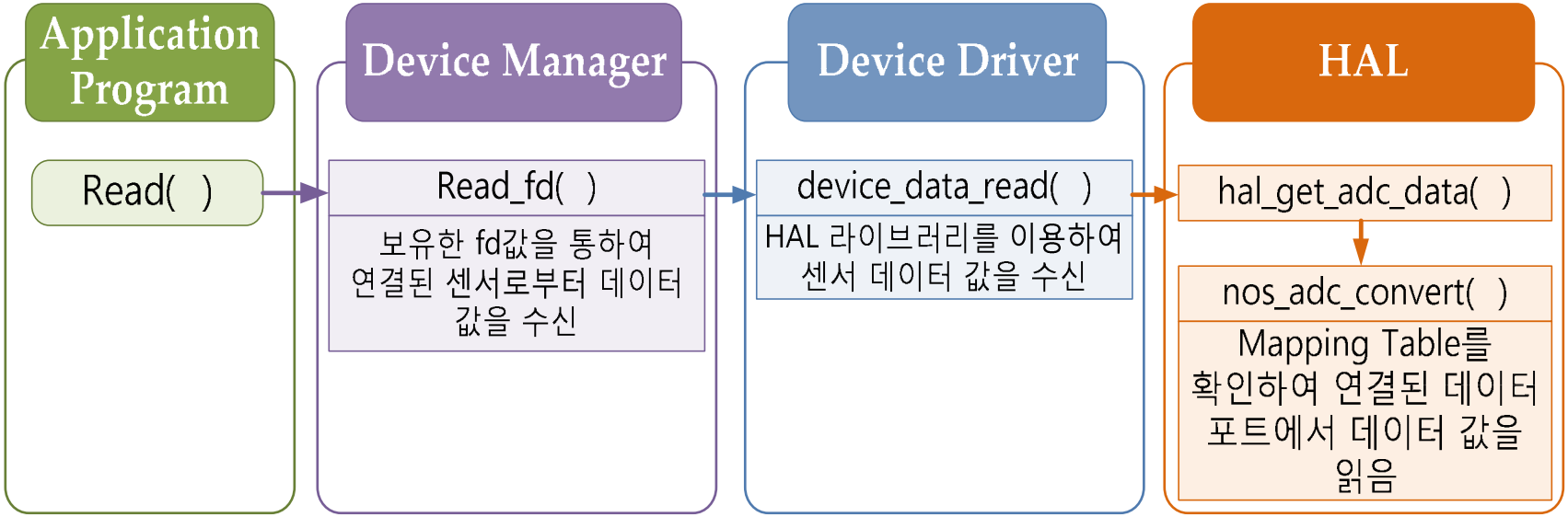
센서 전원 공급 과정



- 응용개발자는 Ioctl()함수를 통하여 연결된 디바이스에 전원을 공급 및 차단이 가능함.

Plug & Play 설계 [5]

센서 데이터 수신 과정



- 응용개발자는 Read()함수를 통하여 연결된 디바이스로부터 센서 데이터 값을 수신함.

응 용 API

응용 프로그램 소스

```
#include "nos.h"

int main (void)
{
    SenDataList dataList;
    nos_init();
    led_on(1);

    uart_printf("\n\n=========================================\nSensor Plug & Play Test\n=========
    delay_ms(1000);

    Open();
    Ioctl();
    while(1)
    {
        delay_ms(2000);// Do sensing eve
        dataList = Read();
        uart_printf("ADC Sensor data : %f\n",dataList.adc_data);
        uart_printf("I2C Sensor data : %f\n",dataList.i2c_data);
        uart_printf("FREQ Sensor data
    }
    delay_ms(1000); // Do sensing every 1 second
    //Close(fd1);
    return 0;
} ? end main ?
```

센서 데이터 값을 가지고 있기 위한 구조체 선언

현재 연결된 센서모듈에게 식별 정보 수신
수신된 식별 정보를 통해 디바이스 드라이버 연결

연결된 센서모듈에게 전원 인가

연결된 센서모듈로부터 데이터 값을 수신

- 응용개발자는 Open(), Ioctl(), Read()함수만을 사용 함으로써 개발 가능
- 다수의 센서 Data는 Read()함수를 통해 구조체 형태로 수신

Device Driver API

ADC 타입 D/D 소스

```
DrvAdcFuncList a9060_func;
UINT8 a9060_func_connect(void)
{
    a9060_func.init = a9060_init;
    a9060_func.read = a9060_data_read;
    deviceDriver[A9060].sensorData.header = 0x35;
    deviceDriver[A9060].sensorData.EPC_manager = 0x0000001;
    deviceDriver[A9060].sensorData.object_class = 0x00000001;
    deviceDriver[A9060].sensorData.serial_number = 0x102030405;
    deviceDriver[A9060].drv_name = "a9060";
    deviceDriver[A9060].drv_func_pport.power_on = a9060_power_on;
    deviceDriver[A9060].drv_func_pport.power_off = a9060_power_off;
    deviceDriver[A9060].drv_func_adc = &a9060_func;
    deviceDriver[A9060].drv_func_intr= NULL;
    deviceDriver[A9060].drv_func_i2c= NULL;
    return 0;
}
UINT8 a9060_init(UINT8 fd)
{
    hal_adc_init();
    return hal_pport_init(fd);
}
UINT8 a9060_power_on(UINT8 fd)
{
    return hal_pport_on(fd);
}

UINT8 a9060_power_off(UINT8 fd)
{
    return hal_pport_off(fd);
}

float a9060_data_read(UINT8 fd)
{
    return hal_get_adc_data(fd)*1.;
}
```

1. 센서 식별코드를 포함하고 있음.
2. 센서 구동 및 데이터 값을 읽어 오기 위한 함수 포인터를 가지고 있음.

ADC관련 레지스터 및 센서 전원 포트 초기화

ADC센서 모듈 전원 ON/OFF

ADC포트로부터 데이터 값을 수신

HAL 영역 구현

Frequency Type 구현

- 물리적인 변화에 따라 센서 파형의 주파수가 변함.
- Timer/Counter를 이용하여 Rising Edge를 검출 하여 데이터 처리
- ATmega128에서 8비트와 16비트 Timer/Counter를 제공
- 16비트 Timer/Counter만 구현 하였음.

```
UINT16 hal_get_frequency_data(UINT8 fd, UINT8 state, UINT16 count_time)
{
    UINT16 freq_data;
    switch((nos_mappingTable+fd)->dataPort.port[0] & 0xFF)  {
    case PORT_D|PIN_6:
        T1_START_Freq(state);
        nos_delay_ms(count_time);
        T1_STOP_Freq(state);
        T1_READ_DATA(freq_data);
        T1_CLEAR_TCNT1();
        return freq_data;
    case PORT_E|PIN_6:
        T3_START_Freq(state);
        nos_delay_ms(count_time);
        T3_STOP_Freq(state);
        T3_READ_DATA(freq_data);
        T3_CLEAR_TCNT3();
        return freq_data;
    default : return 0xFFFF;
    }
}
```

향 후 목 표

완료 사항

- 로컬에서 인식 하는 센서 인터페이스 구현
- 센서 인터페이스에 대한 모듈 설계
  - 센서 식별 정보를 보낼 수 있는 모듈 설계
  - 센서 모듈에 대한 인터페이스 구조 확립
- 센서 식별 정보 전송에 관련한 프로토콜 구현

예정 사항

- 일반 펌웨어 상에 선서 개방형 Plug & Play 구현
- 센서 Plug & Play 실 수행 예제 구현
  - ADC / I2C / Interrupt / SPI Type의 Sensor 예제
- Plug & Play 관련 교육 실시 예정(10월)

Thank You !

http://hci.hannam.ac.kr

HCI Lab.
Hannam University