IoT 기술을 적용한 전신주 안전 관리 시스템
(A UTILITY POLE SAFETY MANAGEMENT SYSTEM USING IOT TECHNOLOGIES)

IoT가전/스마트홈 3팀(박상수, 김진우, 김민경, 사애진, 신예빈)

산업교수 김기화

책임교수 김양중

KOREA POLYTECHNIC UNIVERSITY
한국산업기술대학교

목차

I. 연구 배경

- 한전의 전신주 폴, CCTV용 폴, 신호등 폴, 가로등 폴 및 안내표지 폴 등 다양한 용도로 사용되는 지주들은 장기간 사용에 따른 노후화, 무거운 변압기 설치, 통신 및 전원 케이블의 다중포설, 지진, 태풍, 지반변동, 차량의 충돌 등의 원인에 의해 시공 이후에 기울어짐이 발생되고, 심하게 기울어져 있는 경우, 차량의 **운전자 또는 보행자들에게 위험한 요소로 작용**할 수 있음.

- 지주의 **즉각적인 위험 예방이나 유지보수가 가능하게 함으로써 지주로부터 발생되는 위험을 최소화**가 시급한 실정.

I. 연구 배경

- **지주(폴)**은 일정한 간격으로 어느 지역에나 있기에 **센서를 통해 다양한 정보를 얻을 수 있는 매개체**

- 특히 환경적 데이터를 얻기에 좋은 매개체로 기상 관련 데이터를 수집 및 모니터링 하여 실시간 기상정보를 제공할 수 있음

II. 연구 목표

- IoT 기반의 **지주(폴) 상태정보 검출·진단 시스템 및 모니터링 시스템 개발로 사고예방 및 신속한 유지보수 환경 구축**

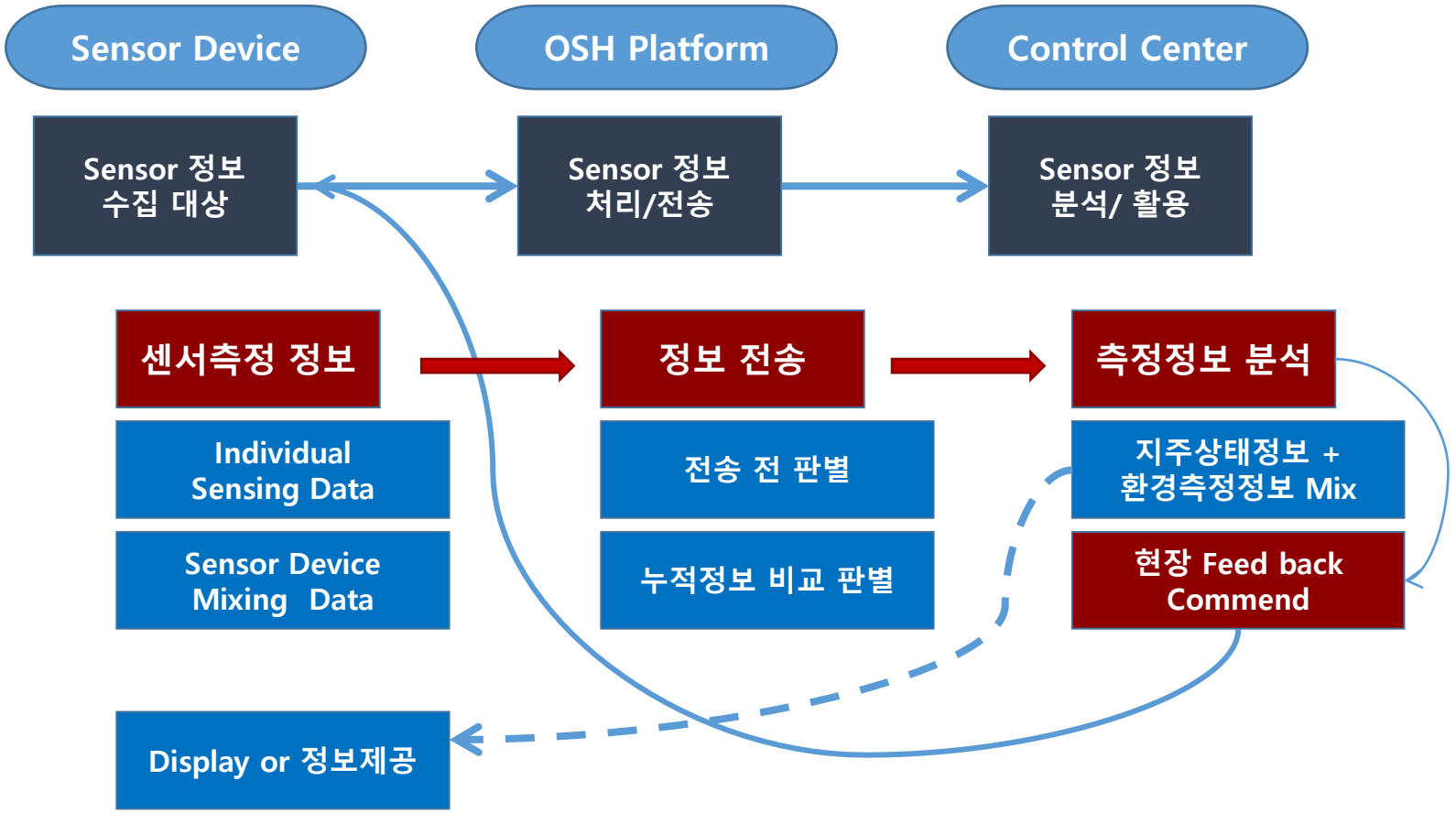
- **기상 정보를 실시간 모니터링 할 수 있는 환경 구축**

III. 연구 내용

- IoT 기반 지주(POLE) 기울기, 충격 측정 장치 개발
- IoT 기반 생활 환경 센서(온/습도, 미세먼지 등) 측정 장치 개발
- 지주 상태 및 환경 측정 정보 실시간 모니터링 시스템 개발

III. 연구 내용 - 주요개발요소

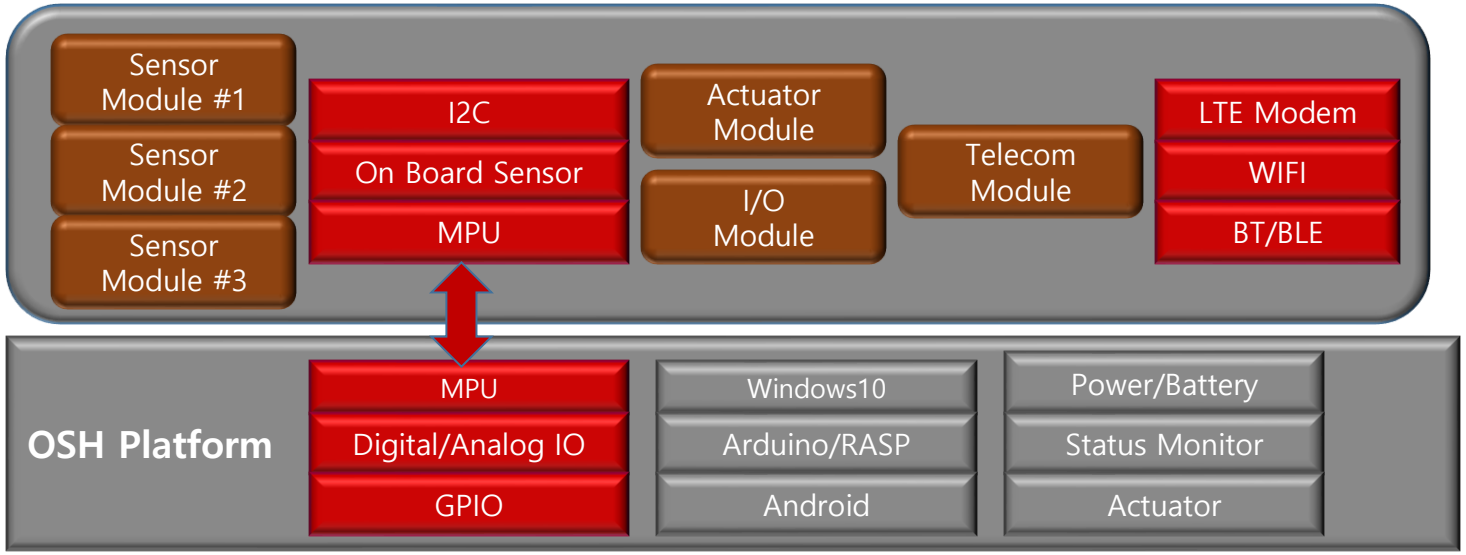
- IoT 기반 지주 안전관리 시스템 운영 시나리오



III. 연구 내용 – Sensor Device 구성

- **Sensor Device Platform(OSHP+Sensor Module)**

  - MPU를 장착한 독립적인 PC Level의 Platform
  - 다양한 Sensor로부터의 정보를 수집 하여 정보를 통합관리
  - 운영자 Server또는 Cloud Base의 DB data 전송 기능
  - OSHP기반의 전용 Platform에 호환이 되는 Sensor 모듈 구현

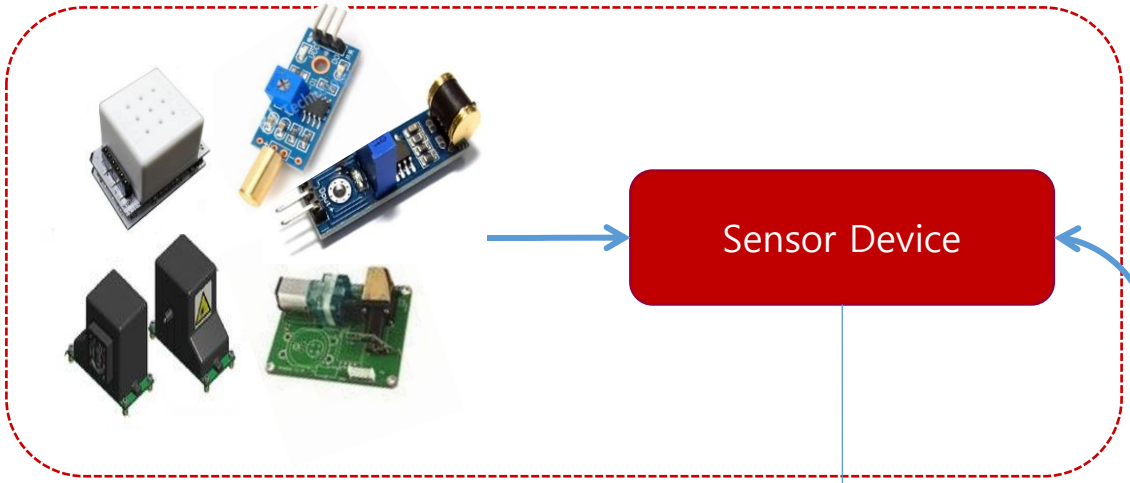


< Sensor Device Platform 구성도>

III. 연구 내용 – 시스템 구성도

- 지주 상태(기울기, 충격), 환경, 기상 정보 등 다양한 센서 측정 모듈 개발

- 정보의 수집에서 모니터링 시스템까지를 일원화하여 정보 활용성 극대화



Sensor Device

Management Server

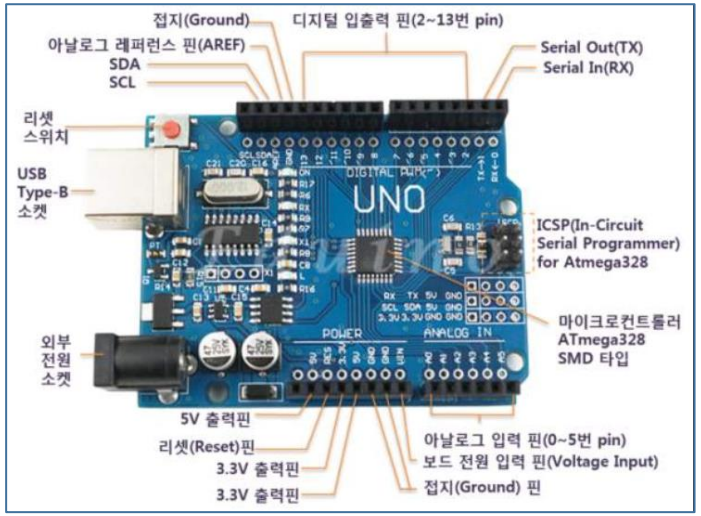
Tele./NW Function | Platform With MCU | Sensor Function | Actuator Function | I/O Function

기울기 센서 | 충격 센서 | 미세먼지 센서 | 온,습도 센서 | 기타 환경 센서

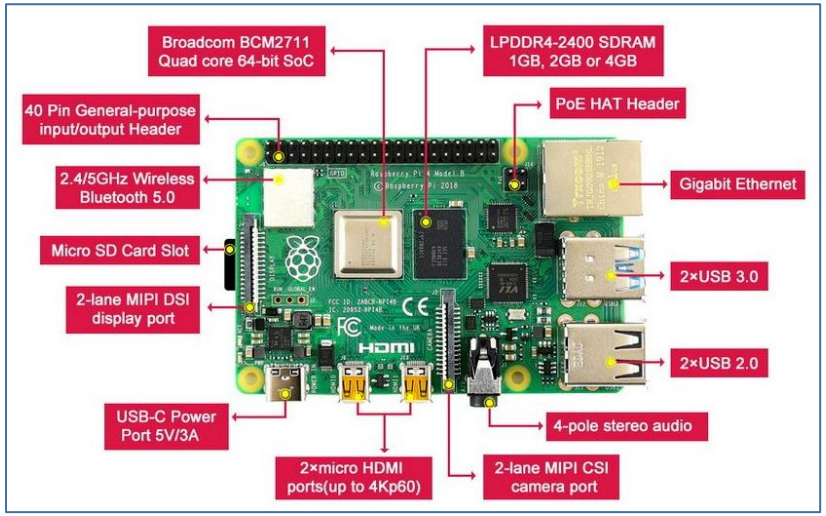
III. 연구 내용 – Sensor Dveice Platform 선택

H/W Platform

- 아두이노
- 라즈베리파이
- Cubie Board



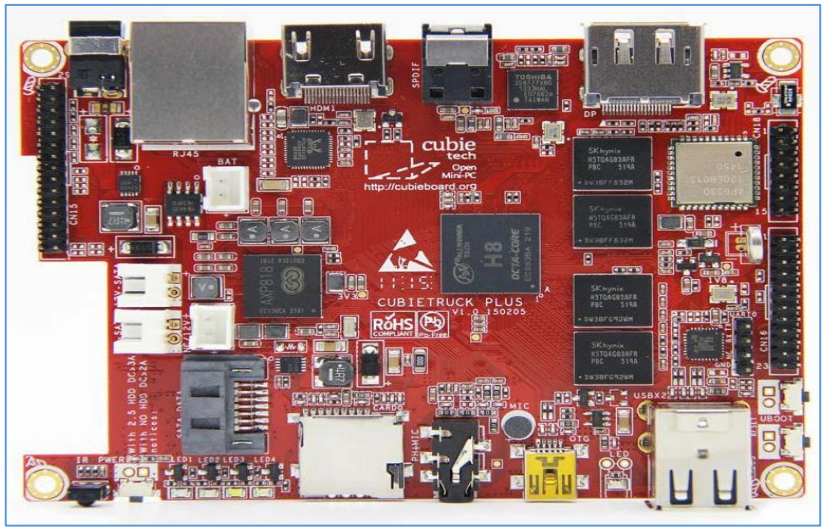
아두이노



라즈베리파이

라즈베리파이 선택 이유

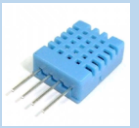
- 세계적으로 많이 판매, 정보와 소스가 많다
- 초기 개발자 접근이 쉽다
- H/W 성능이 본 연구개발 목적에 적합
- 자체 이더넷 통신모듈
- Micro SD card 지원
- Cubie Board 대비 가격이 저렴



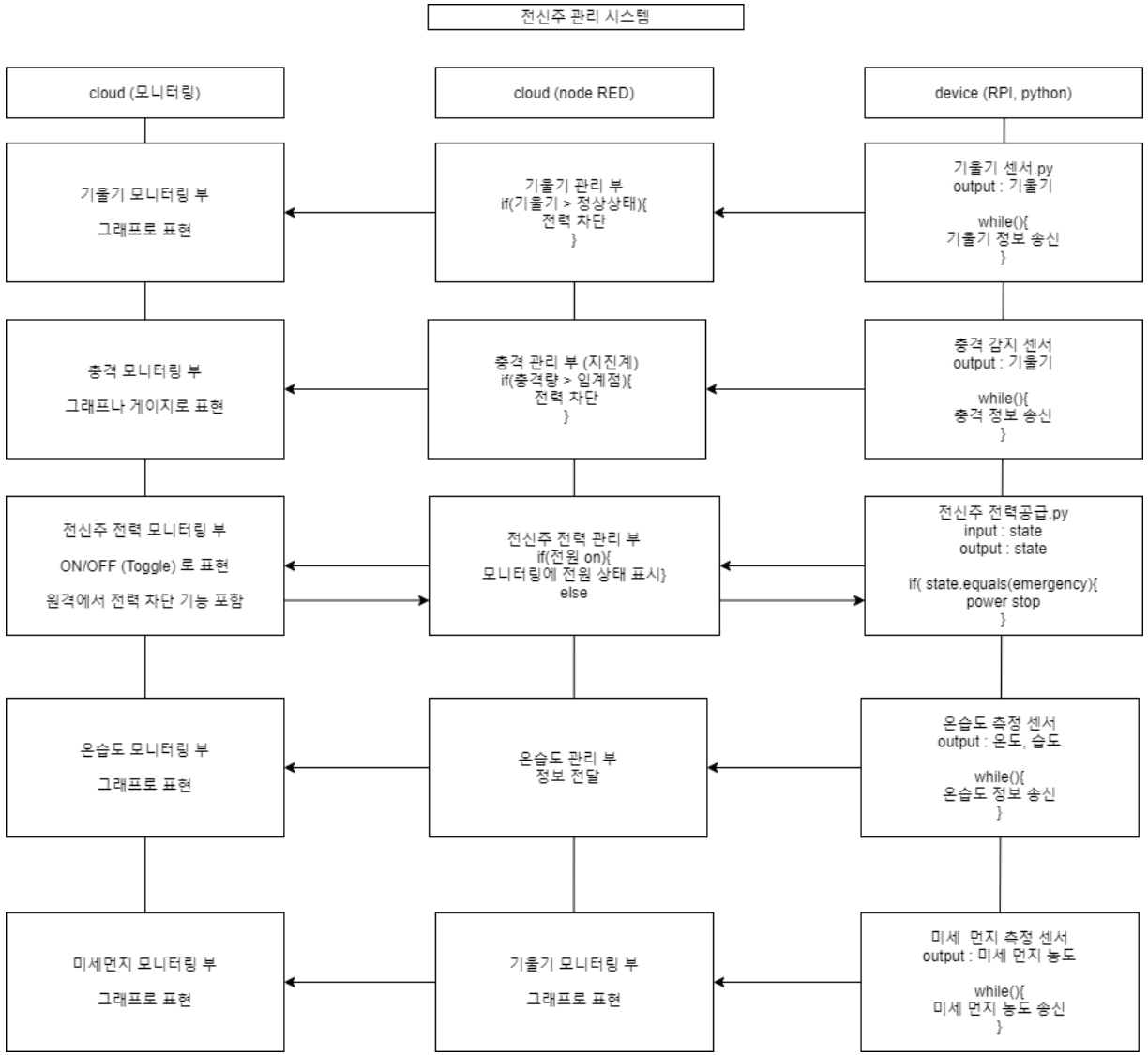
Cubie Board

III. 연구 내용 – Sensor 선택

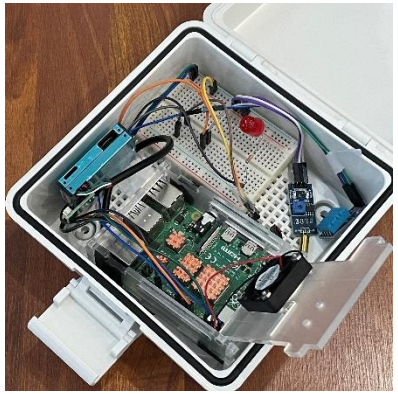
- **Sensor module별 Spec.**

센서 종류		형명	모델명	형태	통신	제조사	구매처	사진
기울기 센서	기울기		SZH-EK084	모듈 (PCB)	serial		device mart	
충격감지 센서	충격		KY-002	모듈 (PCB)	serial	KEYES	device mart	
온습도센서	온습도	C, F	dht11	모듈 (PCB)	Serial	AOSONG	device mart	
먼지센서	먼지	Pm1.0 Pm2.-0 Pm10.0	PM7003	모듈 (PCB)	Serial	PLANTOWER	device mart	

III. 연구 내용 – 시스템 Flow



III. 연구 내용 – 통신 Data 및 Database 구조



IBM Watson IoT

Node-RED

III. 연구 내용 – H/W 개발

하드웨어 부분 개발

라즈베리 파이와 센서를 이용하여 회로를 구성하고 각 센서와 클라우드 플랫폼이 통신할 수 있는 환경을 구축.

III. 연구 내용 – H/W 개발

1. 기울기 센서 제어 및 통신 코드 작성

cloudTiltS.py | crashCloudS.py

```
27
28          deviceCli.publishEvent("status", "json", data, qos=0)
29
30  deviceCli = wiotp.sdk.device.DeviceClient(deviceOptions)
31  deviceCli.commandCallback = commandProcessor
32  deviceCli.connect()
33
34
35  while True:
36
37      result = GPIO.input(2)
38      if result == 1:
39          print("down")
40          tilt = "down"
41          time.sleep(1)
42
43      else:
44          print("low")
45          tilt="low"
46          time.sleep(1)
47
48      data = {"d":{}}
49      data["d"]["inclination"] = tilt;
50
51      deviceCli.publishEvent("status", "json", data, qos=0)
52      sleep(10)
53
```

설명

기울기 값을 센서로부터 받아서 json 형태로 IBM 클라우드 플랫폼에 전송

핵심 코드

```
result = GPIO.input(2)
if result == 1:
tilt = "down"
...
 data = {"d":{}}
 data["d"]["inclination"] = tilt;
 deviceCli.publishEvent("status", "json", data, qos=0)
```

III. 연구 내용 – H/W 개발

2. 충돌 감지 센서 제어 및 통신 코드 작성

cloudTiltS.py | crashCloudS.py

```
        data = {"d":{}}

        data["d"]["cpu_count"] = psutil.cpu_count()
        data["d"]["cpu_freq"] = psutil.cpu_freq().current
        data["d"]["memory"] = psutil.virtual_memory().total

        deviceCli.publishEvent("status", "json", data, qos=0)

deviceCli = wiotp.sdk.device.DeviceClient(deviceOptions)
deviceCli.commandCallback = commandProcessor
deviceCli.connect()


while True:
    result = GPIO.input(3)
    if result != 1:
        impact = 50;
        print("click")
    time.sleep(0.5)

    data = {"d": {}}
    data["d"]["sense_impact"] = impact;

    deviceCli.publishEvent("status", "json", data, qos=0)
    sleep(3)
```

설명

충돌 정보를 센서로부터 입력 받아 json 형태로 클라우드 플랫폼에 전송

핵심 코드

```
result = GPIO.input(3)
if result != 1:
impact = 50;
...
 data = {"d":{}}
 data["d"][“sense_impact"] = tilt;
 deviceCli.publishEvent("status", "json", data, qos=0)
```

III. 연구 내용 – H/W 개발

3. 미세먼지 센서 제어 및 통신 코드 작성

```
def commandProcessor(cmd):
    global switch_state
    if cmd.data["d"]["switch_state"]:
        data = {}
        if cmd.data["d"]["switch_state"] == "on":
            switch_state = 'on'
            GPIO.output(27,True)
            print("Lamp is On")
            data = {"d": {"switch_state": "on"}}
        else:
            switch_state = 'off'
            GPIO.output(27,False)
            print("Lamp is Off")
            data = {"d": {"switch_state": "off"}}
        deviceCli.publishEvent("status", "json", data, qos=0)


deviceCli = wiotp.sdk.device.DeviceClient(deviceOptions)
deviceCli.commandCallback = commandProcessor
deviceCli.connect()


def periodicPublish():
    data = {"d": {"switch_state": switch_state}}
    deviceCli.publishEvent("status", "json", data, qos=0)
    print("        Periodic update : Lamp is " + switch_state)


while True:
```

설명

미세먼지의 입자의 크기에 따라 구분하여 각각을 IBM 클라우드 플랫폼에 전송

핵심 코드

```
dust.print_serial(buffer)
...

data = {"d":{}}
data["d"][“pm_1.0"] = pm1;
data["d"][“pm_2.5"] = pm2;
data["d"][“pm_10.0"] = pm10;

 deviceCli.publishEvent("status", "json", data, qos=0)
```

III. 연구 내용 – H/W 개발

4. 온습도 센서 제어 및 통신 코드 작성

cloudTiltS.py | crashCloudS.py | dustSens.py

```
150
151  # USE PORT
152  SERIAL_PORT = UART
153
154  # Baud Rate
155  Speed = 9600
156
157  # example
158  if __name__ == '__main__':
159
160      # serial setting
161      ser = serial.Serial(SERIAL_PORT, Speed, timeout=1)
162
163      dust = PMS7003()
164
165      while True:
166
167          ser.flushInput()
168          buffer = ser.read(1024)
169
170          if (dust.protocol_chk(buffer)):
171
172              print("DATA read success")
173
174              # print data
175              dust.print_serial(buffer)
176
177          else:
```

설명

현재 전력 공급 상태를 나타내는 LED의 켜짐/꺼짐 정보를 실시간으로 클라우드에 전송하며, 클라우드에서 전원 제어 명령이 내려오면 그에 맞게 전원 공급/차단

핵심 코드

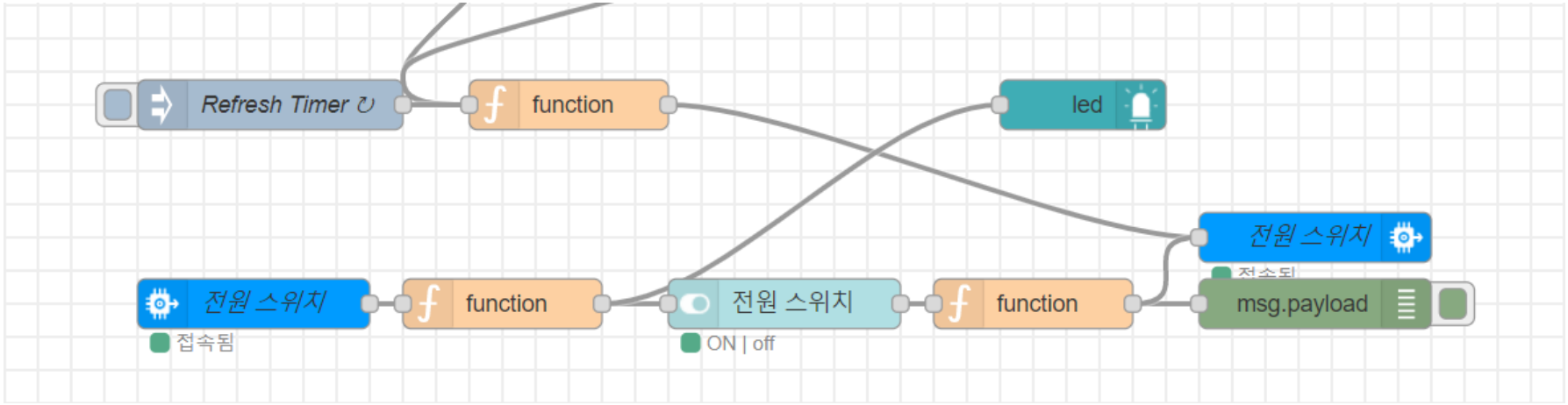
```
def commandProcessor(cmd):
    global switch_state

if cmd.data["d"]["switch_state"] == "on":
            switch_state = 'on'
            GPIO.output(27,True)
            print("Lamp is On")
            data = {"d": {"switch_state": "on"}}
deviceCli.publishEvent("status", "json", data, qos=0)
```

III. 연구 내용 – Monitoring s/w 개발

- IoT 클라우드 플랫폼을 이용하여 디바이스와 통신 하기 위한 각각의 디바이스 연동
- 디바이스 이벤트들을 관리, Node-Red를 이용하여 디바이스 이벤트에 대하여 상황에 맞는 알고리즘을 개발
- 모니터링 User Interface 화면 개발

III. 연구 내용 – Monitoring s/w 개발

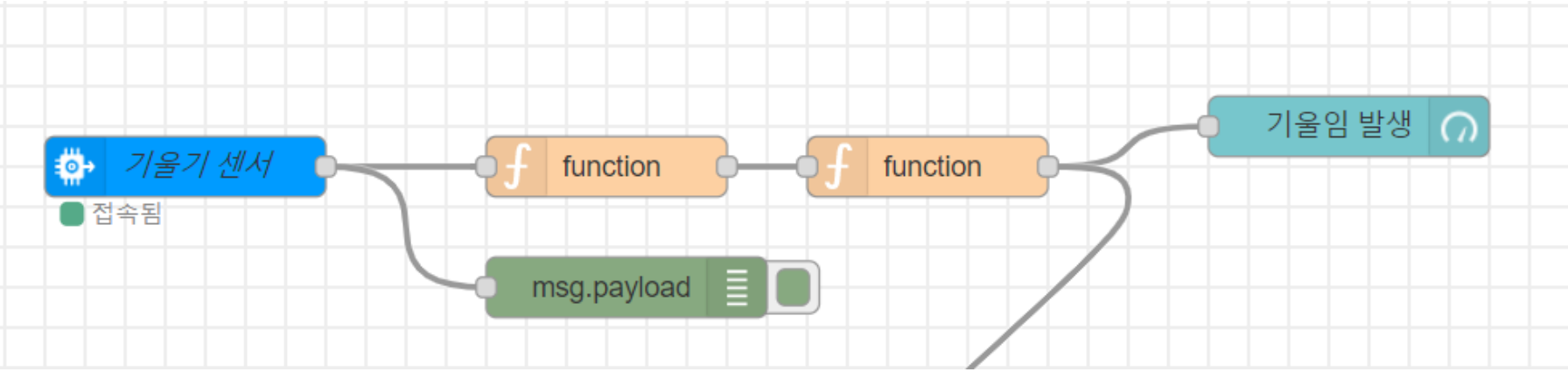
1. 전원 상태 모니터링 및 제어 코드 : 충격, 기울임 등의 이벤트 발생시 자동으로 전신주의 전력을 차단




```
1  var blockd1 = global.get("impact_block"); // 충격감지
2  var blockd2 = global.get("inc_state");  // 기울기
3
4  var evt1={'d':{}};
5
6  if(blockd1 == "block" || blockd2 == "block"){
7      evt1.d.switch_state='off';
8  }else{
9      evt1.d.switch_state='on';
10 }
11
12 return {payload:JSON.stringify(evt1)};
13
```

III. 연구 내용 – Monitoring s/w 개발

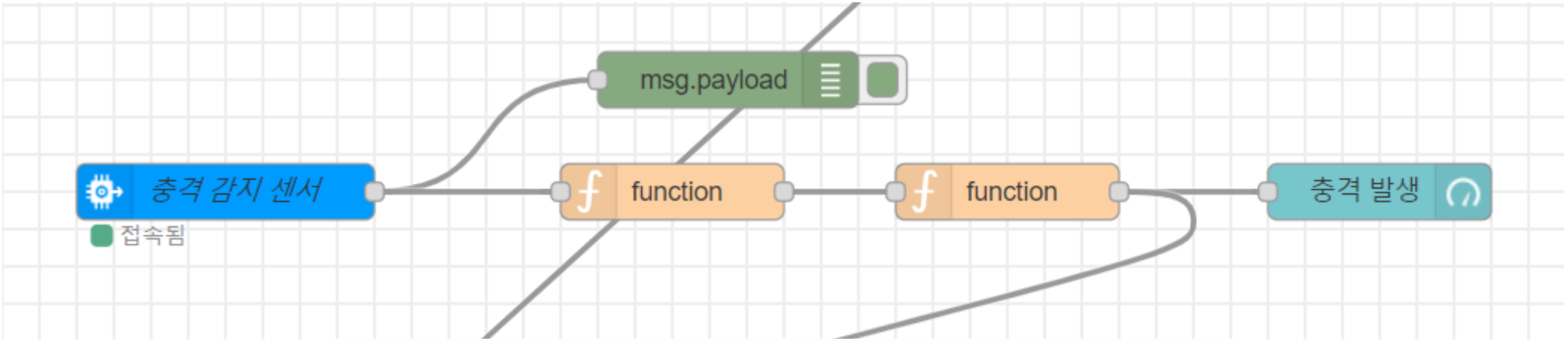
2. 기울기 센서 수신 및 처리 알고리즘 : 기울임이 발생할 경우 알림을 보냄




```
var inc = global.get("inclination");
var state = "on";
     //미래  현재

if( (inc == "0") ){
    state = "unblock";
}
else{
    state="block";
}



msg.payload = inc;
global.set("inc_state",state);

return msg;
```

III. 연구 내용 – Monitoring s/w 개발

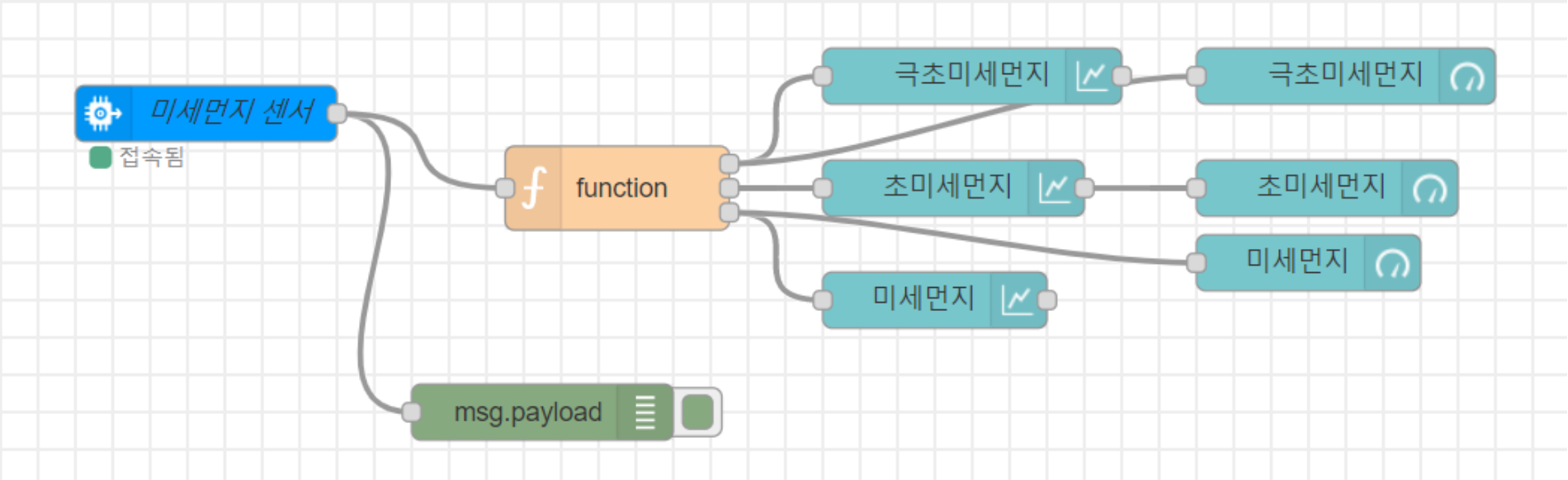
3. 충격 감지 센서 수신 및 처리 알고리즘 : 충격이 발생할 경우(임계치 기준) 알림이 발생




```
1  var sense_impact = global.get("sense_impact");
2  var critical_point = 23;
3  var block = "block";
4
5  if( (sense_impact > critical_point) ){
6      block = "block";
7
8  }
9  else{
10     block = "unblock"
11 }
12
13 msg.payload = sense_impact;
14 global.set("impact_block",block);
15
16 return msg;
```

III. 연구 내용 – Monitoring s/w 개발

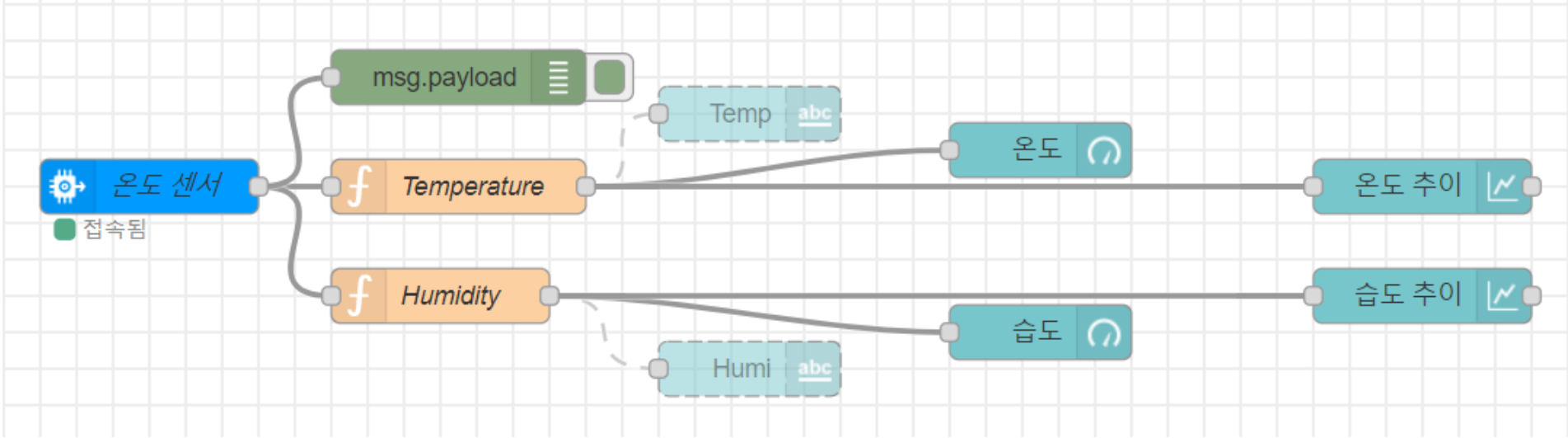
4. 미세먼지 센서 수신 및 처리 알고리즘 : 극초미세먼지, 초미세먼지, 미세먼지 정보 처리




```
1  var pm1 = {payload:msg.payload.d.pm1};
2  var pm2 = {payload:msg.payload.d.pm2};
3  var pm3 = {payload:msg.payload.d.pm3};
4
5  return [pm1,pm2,pm3];
```

III. 연구 내용 – Monitoring s/w 개발

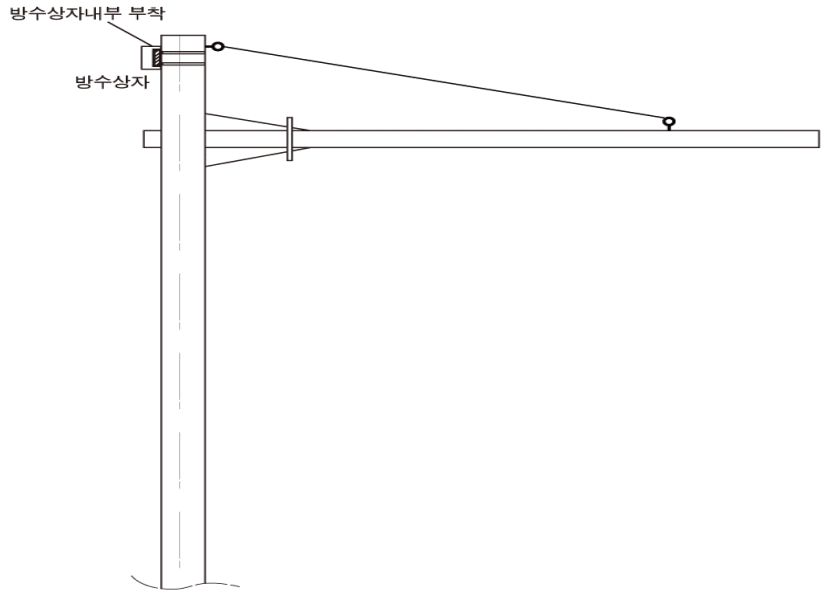
5. 온습도 센서 수신 및 처리 알고리즘 : 온도/습도 정보 처리

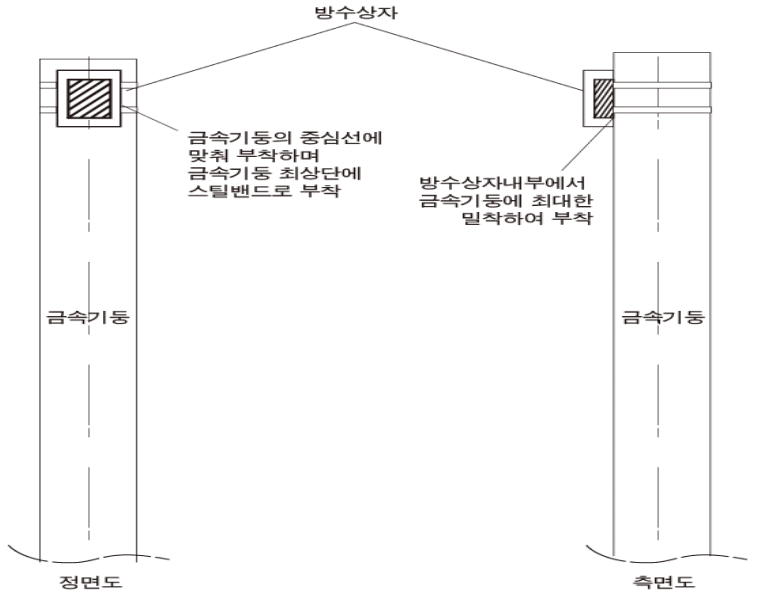


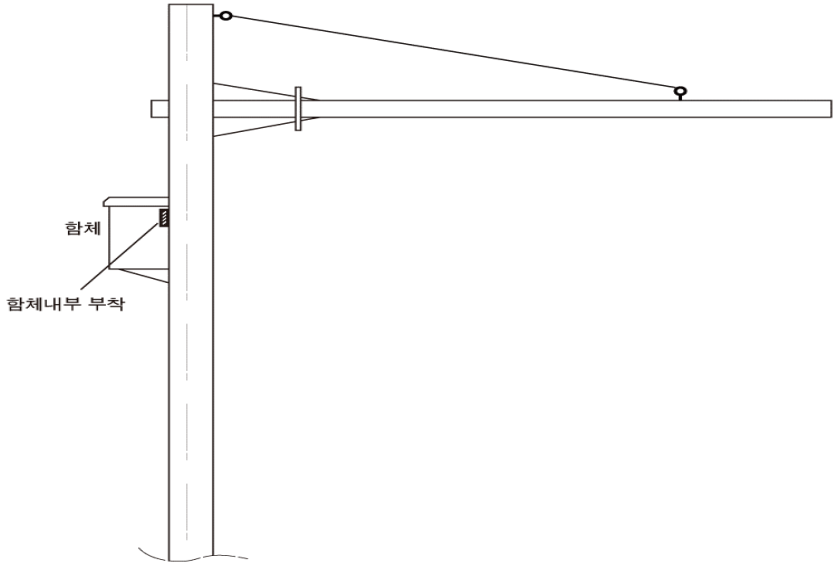

```
1  msg.payload=msg.payload.d.temp;
2  return msg;
```

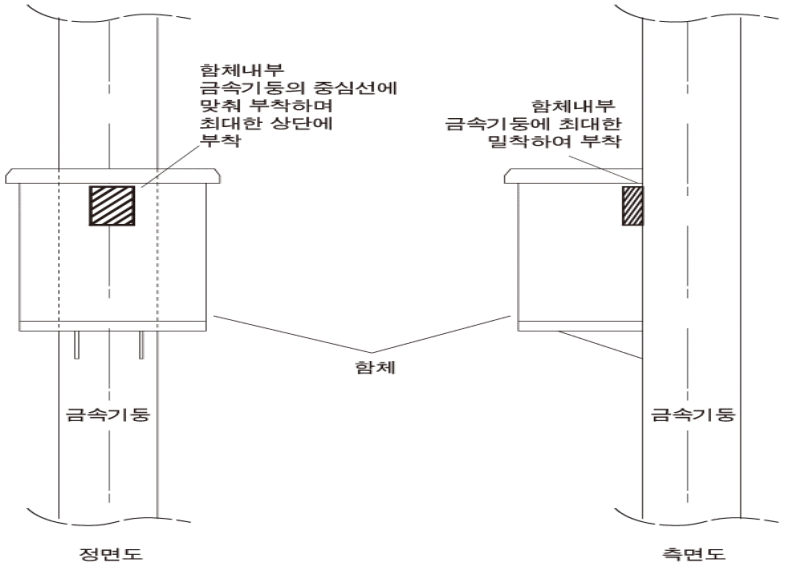
```
1  msg.payload=msg.payload.d.humi;
2  return msg;
```

III. 연구 내용 – 설치 예시

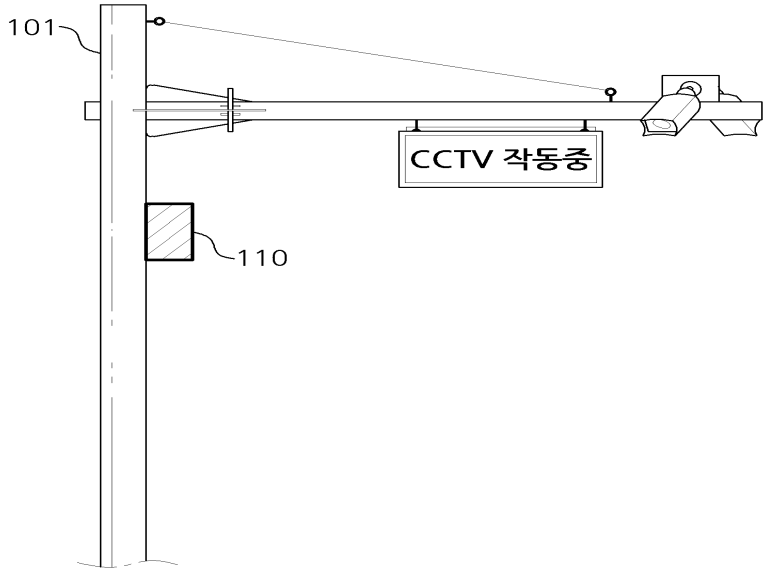


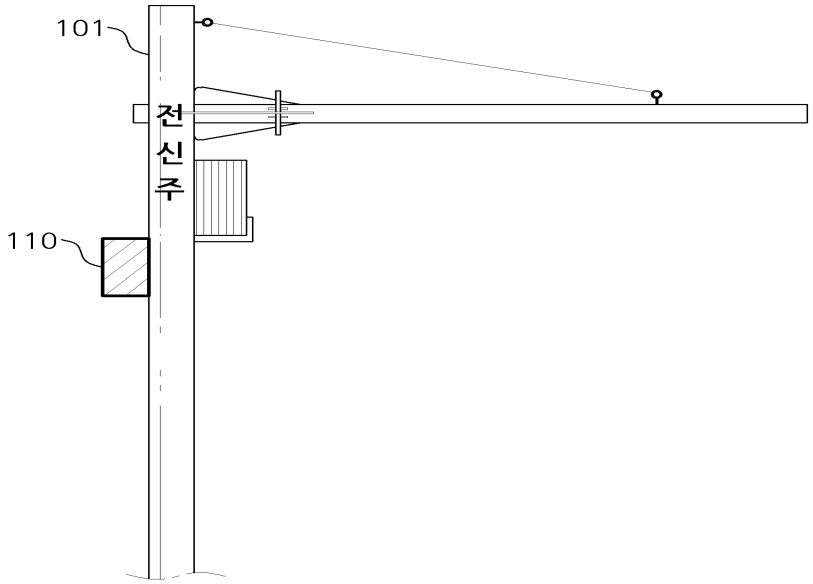


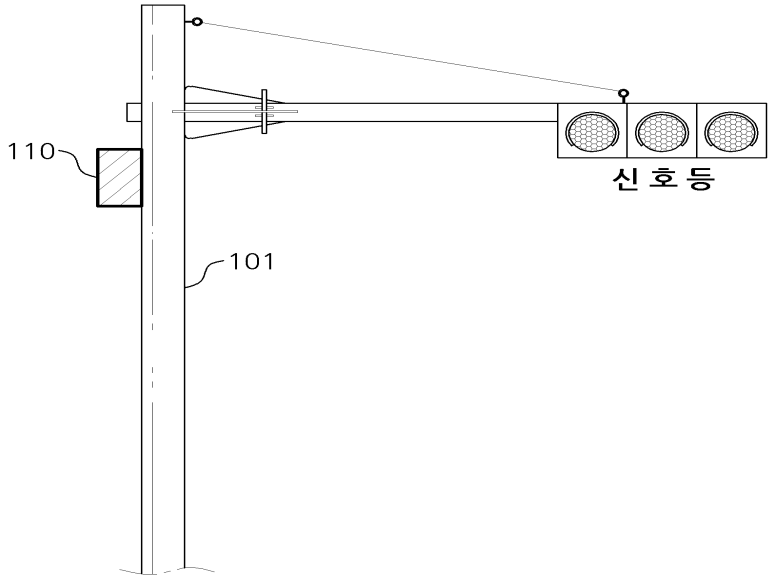


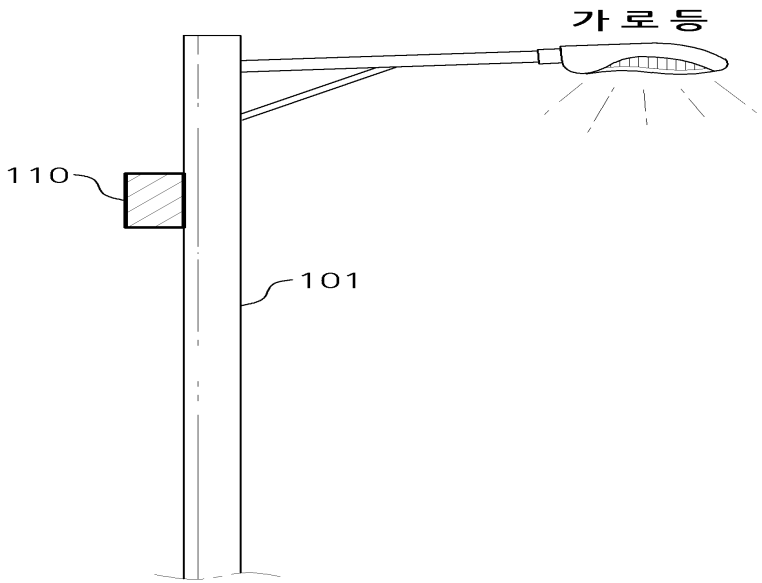


III. 연구 내용 – 설치 예시









III. 연구 내용 – 결과

현재 개발 결과

1) 지주의 기울기 정도확인
2) 외부 충격의 강도 확인
3) 온/습도 수치 확인
4) 미세먼지 농도 수치 확인
5) 이벤트에 대한 전력 차단 기능

향후 필요한 개발 내용

1) 데이터 베이스 구축을 통한 정보 관리 시스템 개발 필요
2) Map을 활용한 지역별 지주현황 모니터링 필요
3) 지속적인 개발을 통한 각 센서 측정값의 세분화, 환경 오염도 정보 누적, 비교 필요
4) 기술융합에 의한 활용성 필요 (CCTV 카메라, 소리 감지 등)

IV. 기대효과 – 관련시장규모 및 사업성

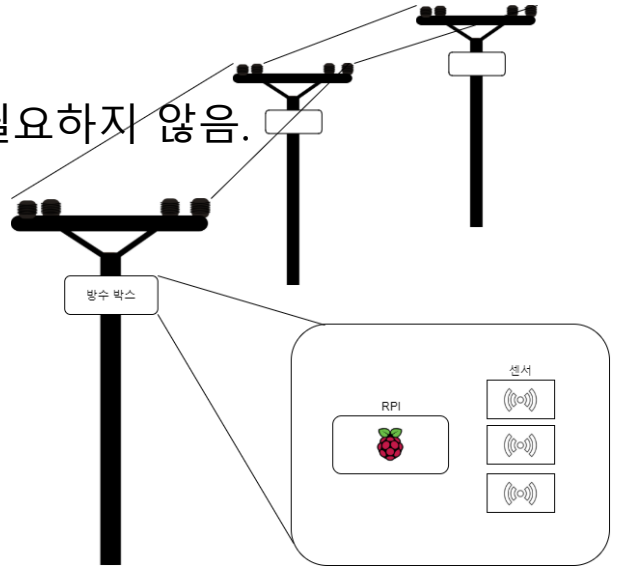
- **다양하고 대량의 정보 수집 및 모니터링 용이**

- **불필요한 건설 비용 감축 효과**

  - 기존의 기상 관측소는 한정된 구역에만 설치되어 있음.

  - 외부 기상 관측소를 건설하지 않고도 전신주에 설치된 모듈을 통해 다양한 지역 내의 많은 기상 정보를 수집할 수 있음.

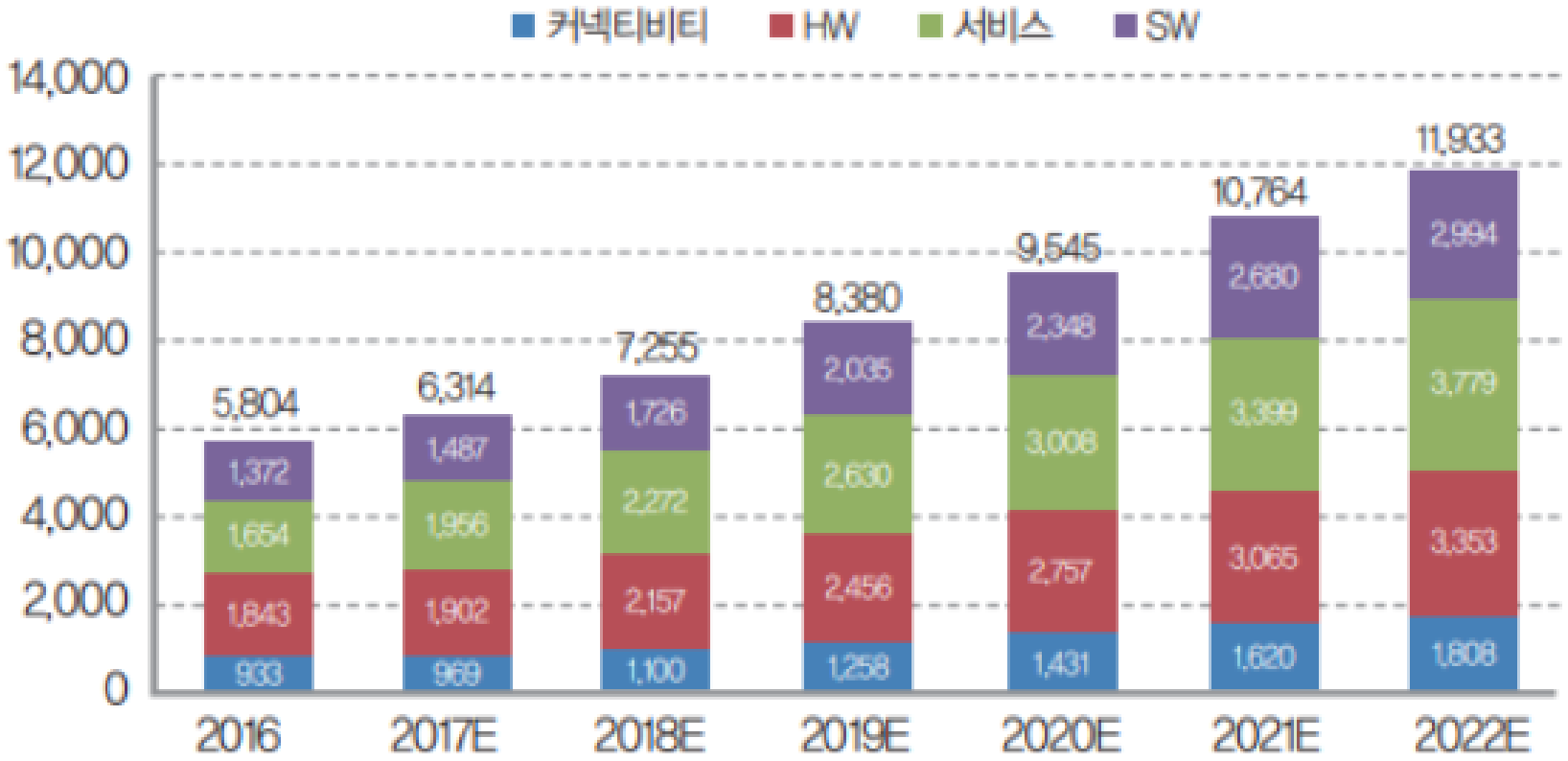
  - 전신주에서 바로 전력을 공급받을 수 있으므로 전력 공급을 위한 추가적인 자원이 필요하지 않음.



Ⅳ. 기대효과 – 관련시장규모 및 사업성

- **세계 IoT 시장규모**

2018년 7,255억 달러로 전년 대비 14.9% 성장했으며, 2016~2022년까지 연평균 12.8% 성장하면서 1조 1,933억 달러에 달할 것으로 전망



(자료원: IDC, 2018,10)

Ⅳ. 기대효과 - 시장현황

- **국내외 IoT 시장 규모**

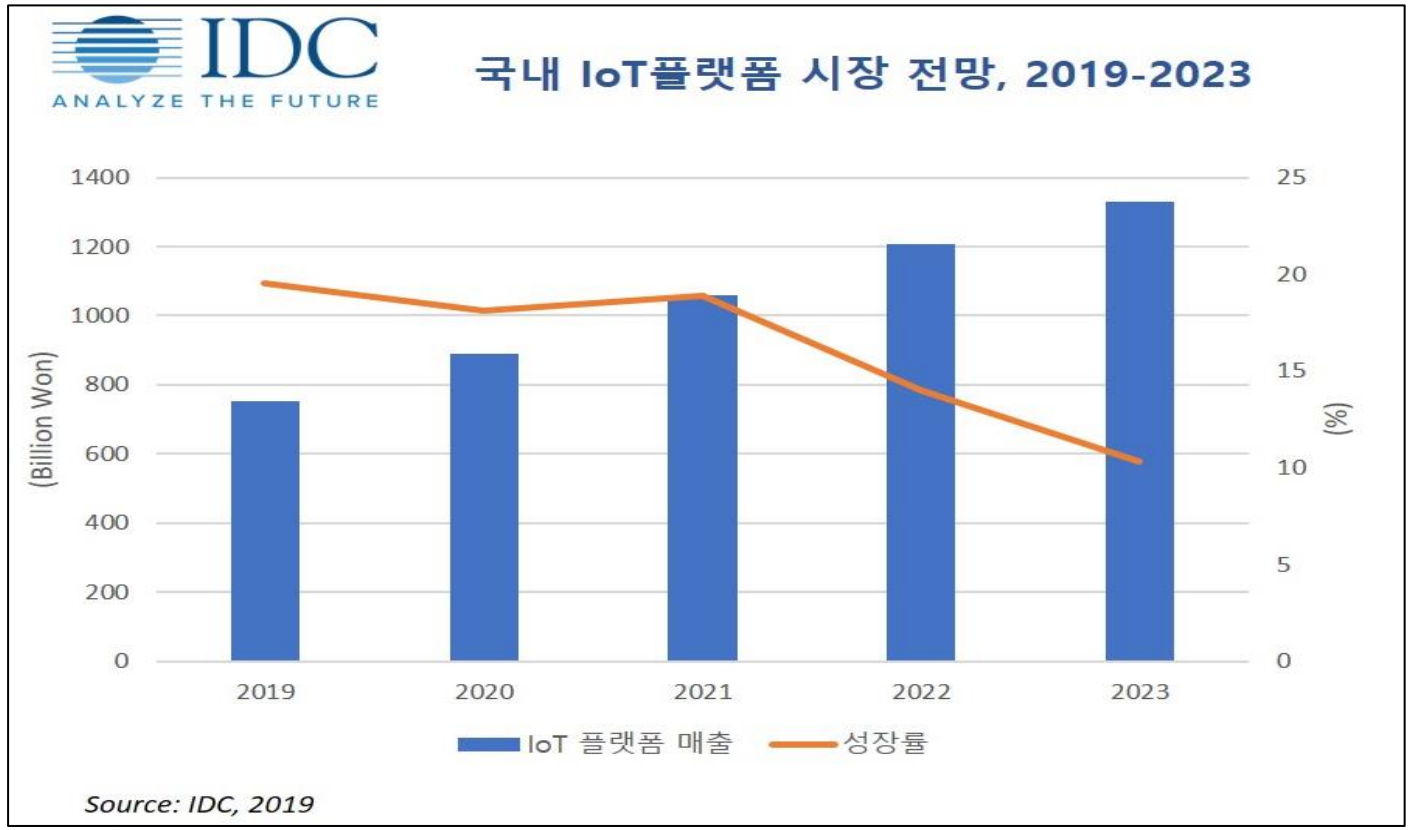
  2022년 세계 IoT 시장 규모는 1225조. 국내 IoT 시장 규모는 약 23조 전망

※ 산업연구원(2014)

IV. 기대효과 – 관련시장규모 및 사업성

- **국내 IoT 플랫폼 시장규모**

2019년 국내 IoT 플랫폼 시장 규모는 전년 대비 19.5% 증가한 7,540억원.
2023년까지 16.1%의 연평균성장률을 보이며 1조 3,308억원에 이를 것으로 전망

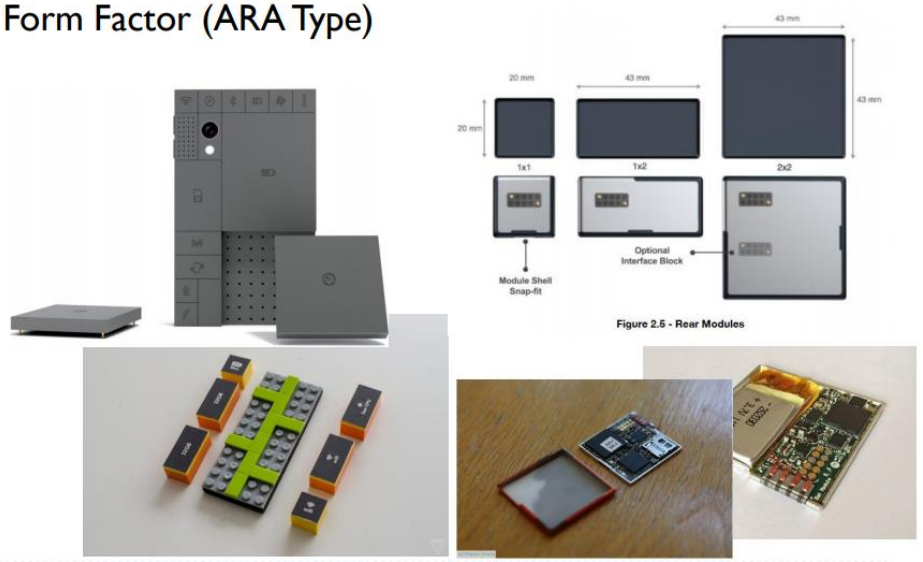


IV. 기대효과 – 제품화 기획

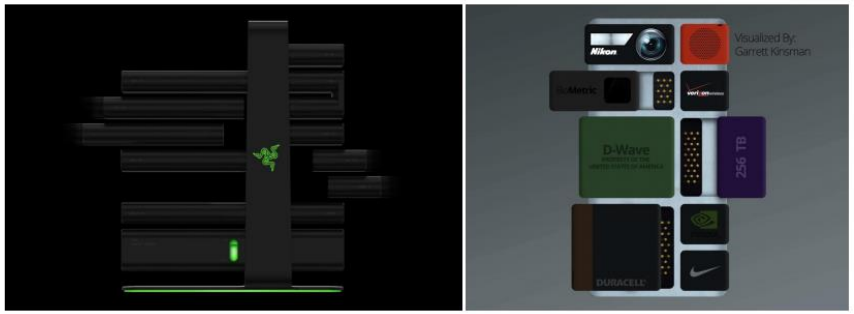
- **Form Factor**

▸ Form Factor (USB Type)

▸ Form Factor (ARA Type)



▸ Form Factor (USB + ARA Type)



감사합니다