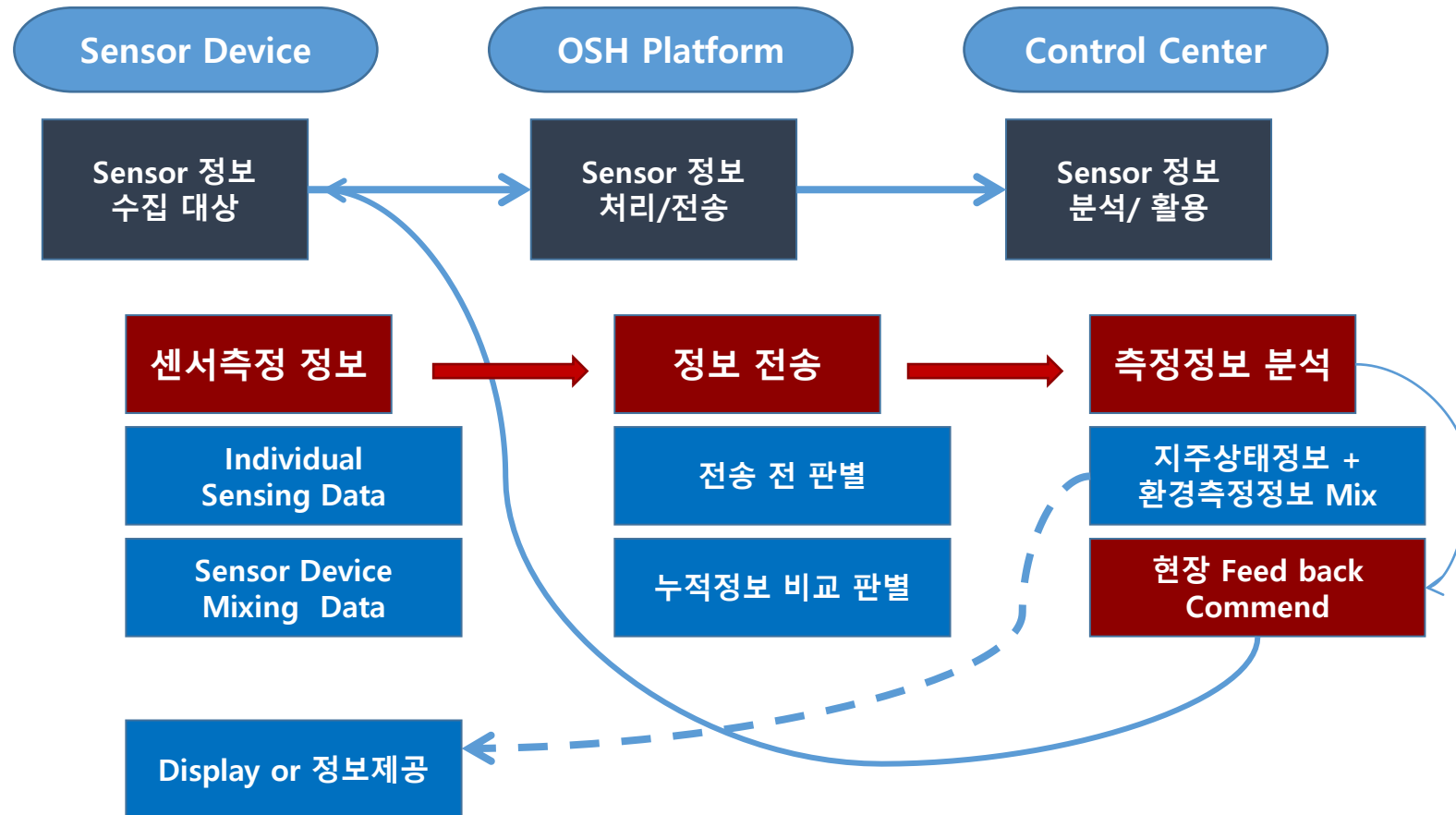


III. 연구 내용 - 주요개발요소

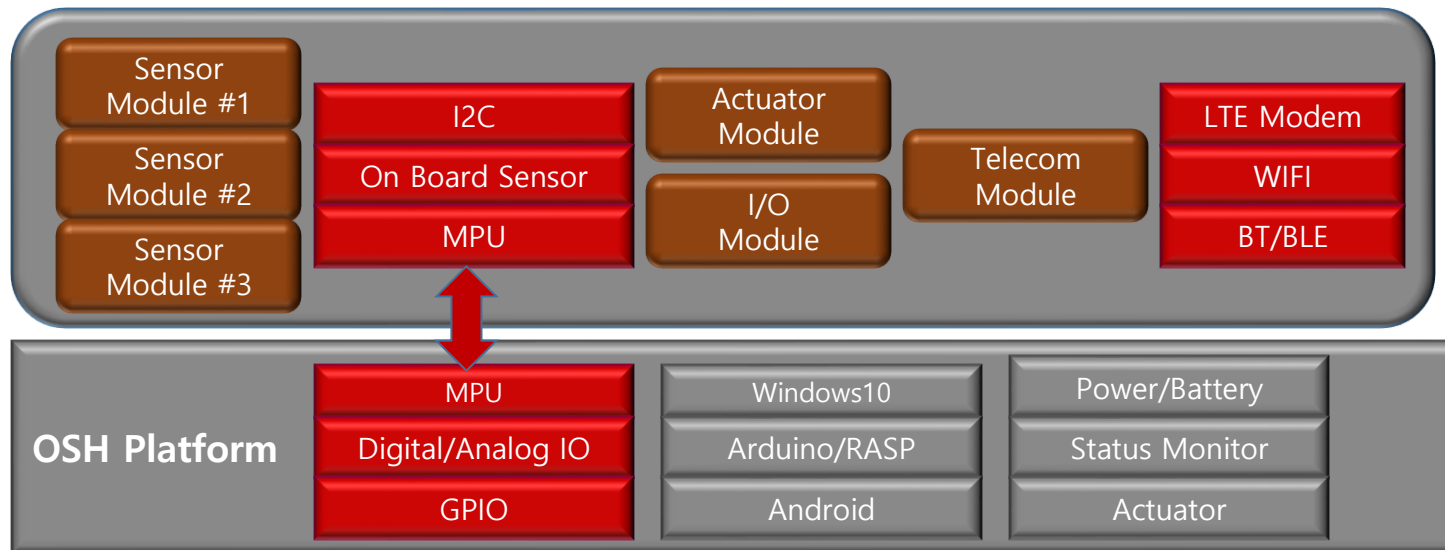
• IoT 기반 지주 안전관리 시스템 운영 시나리오



III. 연구 내용 – Sensor Device 구성

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

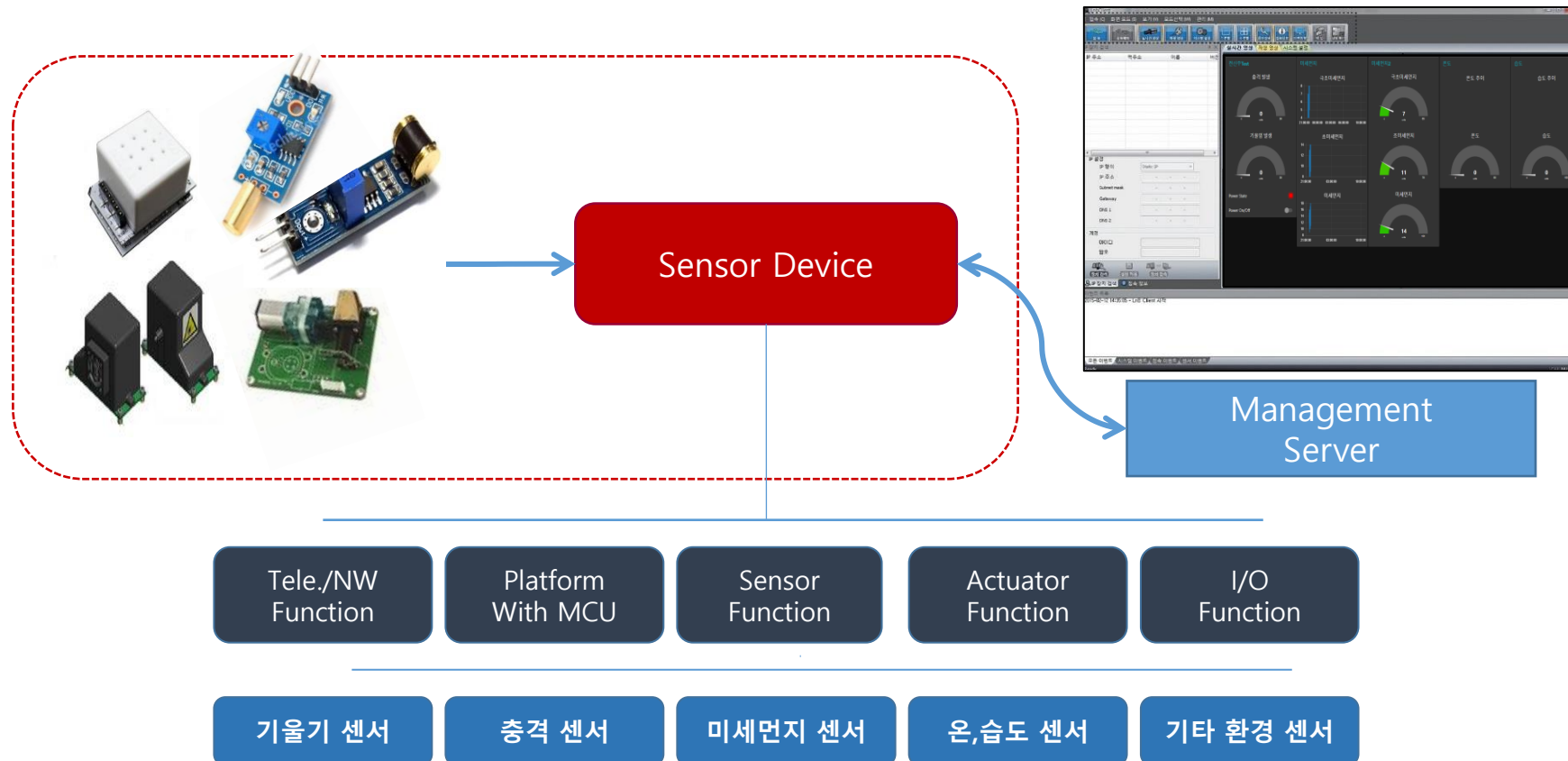
- MPU를 장착한 독립적인 PC Level의 Platform
- 다양한 Sensor로부터의 정보를 수집 하여 정보를 통합관리
- 운영자 Server또는 Cloud Base의 DB data 전송 기능
- OSHP기반의 전용 Platform에 호환이 되는 Sensor 모듈 구현



< Sensor Device Platform 구성도 >

III. 연구 내용 – 시스템 구성도

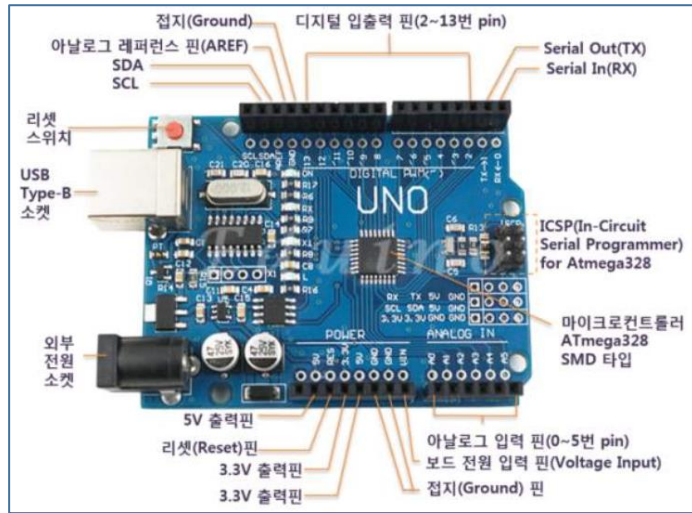
- 지주 상태(기울기, 충격), 환경, 기상 정보 등 다양한 센서 측정 모듈 개발
- 정보의 수집에서 모니터링 시스템까지를 일원화하여 정보 활용성 극대화



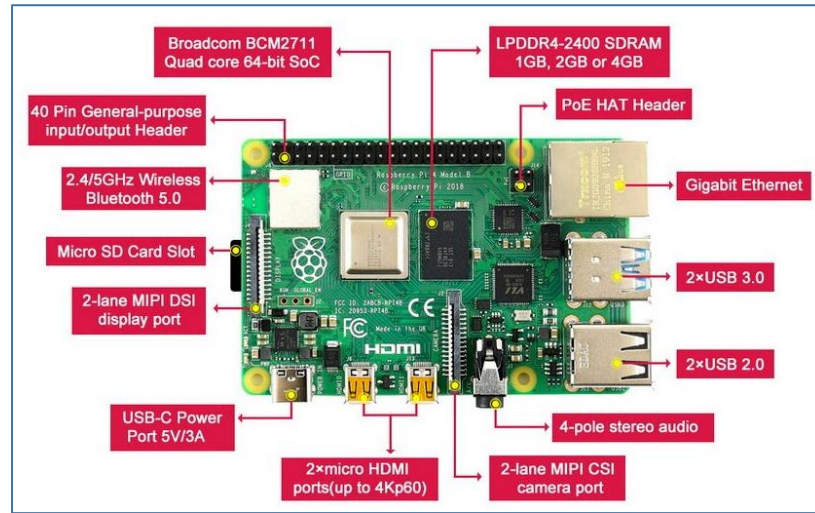
III. 연구 내용 – Sensor Dveice Platform 선택

H/W Platform

- 아두이노
- 라즈베리파이
- Cubie Board



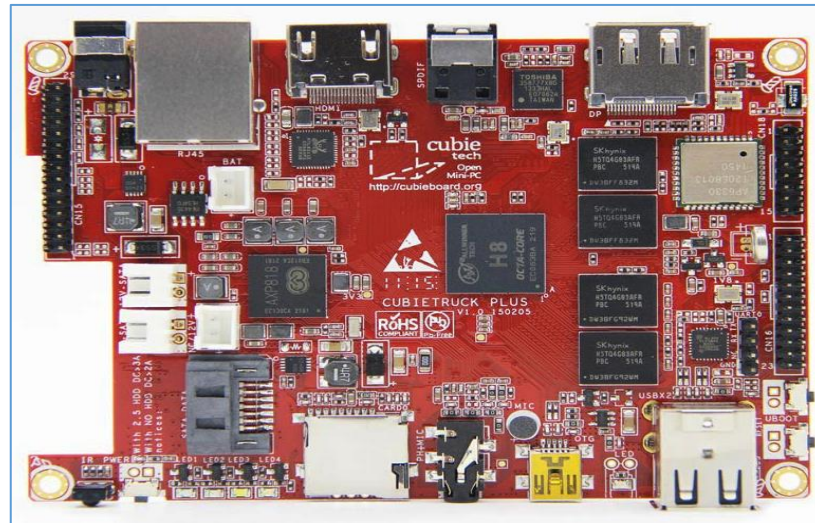
아두이노



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



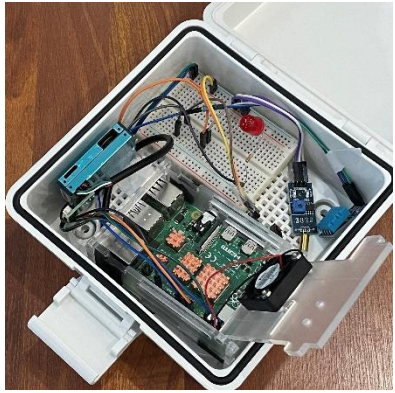
Cubie Board

III. 연구 내용 – Sensor 선택

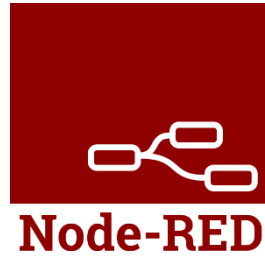
• Sensor module별 Spec.

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 – 통신 Data 및 Database 구조



IBM **Watson IoT**



III. 연구 내용 – H/W 개발

하드웨어 부분 개발

라즈베리 파이프와 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

```
cloudTilt.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 x crashCloudS.py x
22 data = {"d":{}}
23
24 data["d"]["cpu_count"] = psutil.cpu_count()
25 data["d"]["cpu_freq"] = psutil.cpu_freq().current
26 data["d"]["memory"] = psutil.virtual_memory().total
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     result = GPIO.input(3)
37     if result != 1:
38         impact = 50;
39         print("click")
40         time.sleep(0.5)
41
42     data = {"d": {}}
43     data["d"]["sense_impact"] = impact;
44
45     deviceCli.publishEvent("status", "json", data, qos=0)
46     sleep(3)
47
```

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

```
18 def commandProcessor(cmd):
19     global switch_state
20     if cmd.data["d"]["switch_state"]:
21         data = {}
22         if cmd.data["d"]["switch_state"] == "on":
23             switch_state = 'on'
24             GPIO.output(27,True)
25             print("Lamp is On")
26             data = {"d": {"switch_state": "on"}}
27         else:
28             switch_state = 'off'
29             GPIO.output(27,False)
30             print("Lamp is Off")
31             data = {"d": {"switch_state": "off"}}
32         deviceCli.publishEvent("status", "json", data, qos=0)
33
34
35 deviceCli = wiotp.sdk.device.DeviceClient(deviceOptions)
36 deviceCli.commandCallback = commandProcessor
37 deviceCli.connect()
38
39
40 def periodicPublish():
41     data = {"d": {"switch_state": switch_state}}
42     deviceCli.publishEvent("status", "json", data, qos=0)
43     print("    Periodic update : Lamp is " + switch_state)
44
45
46 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

```
cloudTilt.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)
```