

# IoT 스마트팜 시스템에 AI 자동관수제어를 적용하기 위한 연구

지도교수 : 권용진 교수님  
2012122324 호수영

# 목차

---

- ▶ 연구배경과 목표
- ▶ 연구진행사항
  - ▶ 연구실의 기존 스마트팜 시스템 학습
    - ▶ 스마트팜에 사용된 센서학습
    - ▶ 릴레이, 투광기와 솔밸브를 이용한 원격제어 학습
    - ▶ 데이터 축적,확인,원격제어를 위한 서버 구성학습
  - ▶ AI자동관수제어 연구
- ▶ 추후연구계획

# 연구배경과 목표(1/3) -연구배경

- ▶ 최근 현대인들에게 탈도시를 위한 귀농이 유행함
  - ▶ 조용한 전원생활
  - ▶ 도시생활에 대한 회의감
  - ▶ 은퇴 후 여가 생활
- ▶ 초보 귀농인들은 농사에 대한 정보, 관리방법에 대한 습득이 미흡한 문제들이 나타남

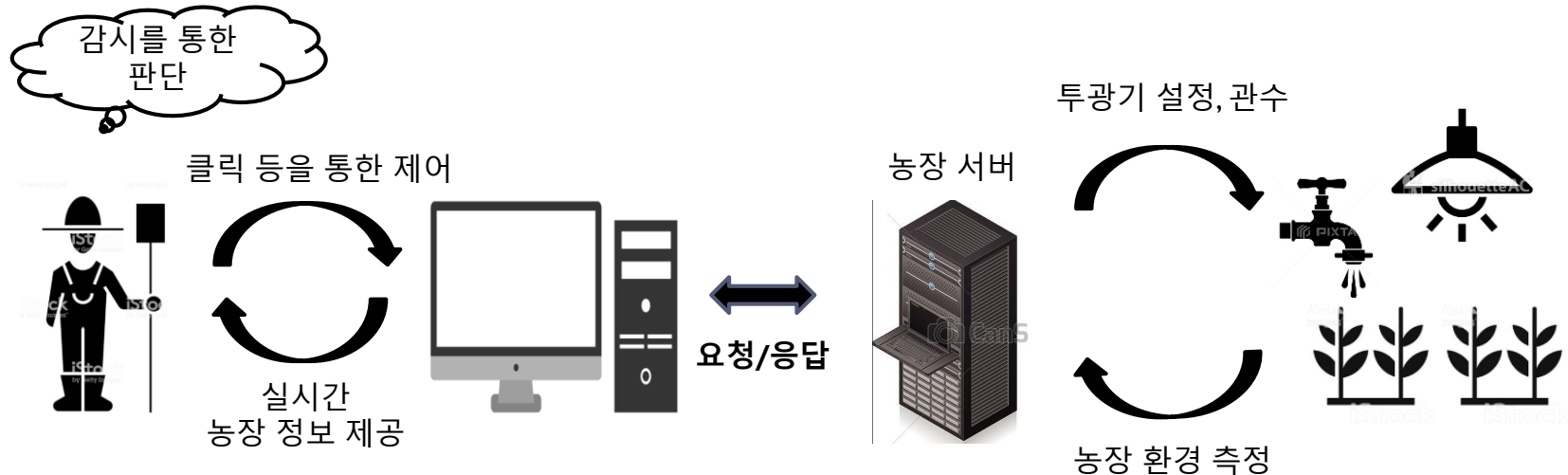


## 귀농·귀촌 장애요인 (복수응답)



## 연구배경과 목표(2/3) -연구배경

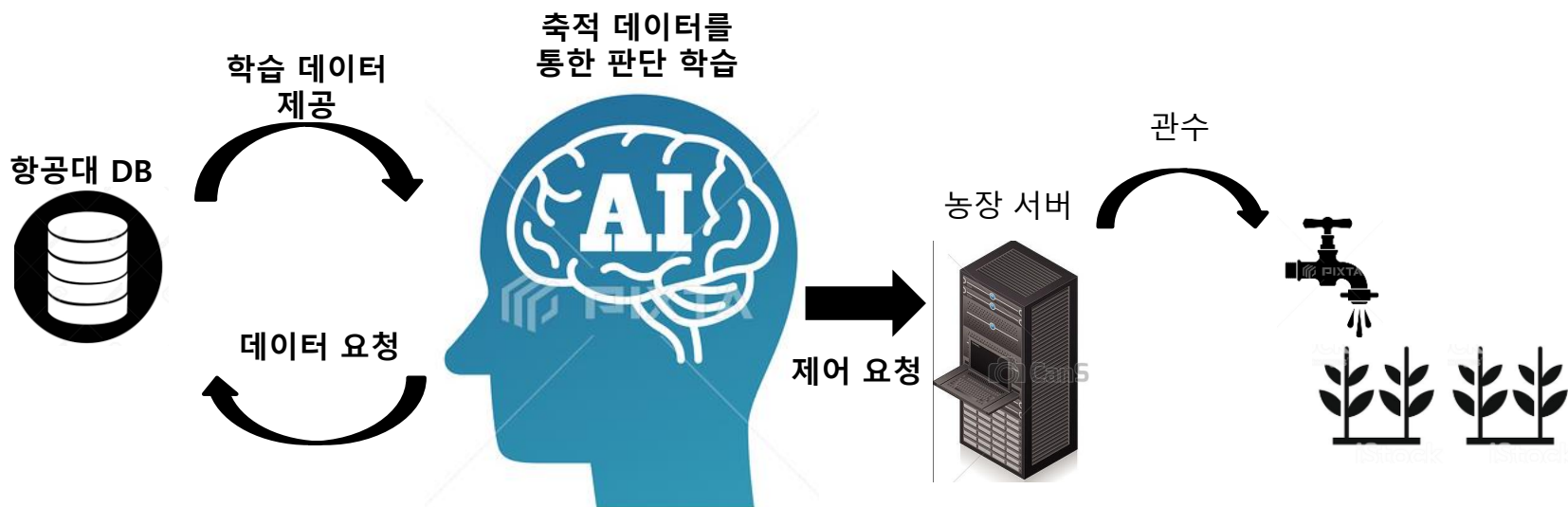
- ▶ IoT 스마트팜 시스템 연구로 해결점 제시
  - ▶ 사용자가 웹 브라우저를 통해 농장정보확인 및 원격제어
  - ▶ 그러나 각 데이터의 의미와 적절한 관수시기판단은 귀농인들에게 어려운 문제이며, 모니터링을 통해 관수를 지시해야하는 한계성



# 연구배경과 목표(3/3) -연구목표

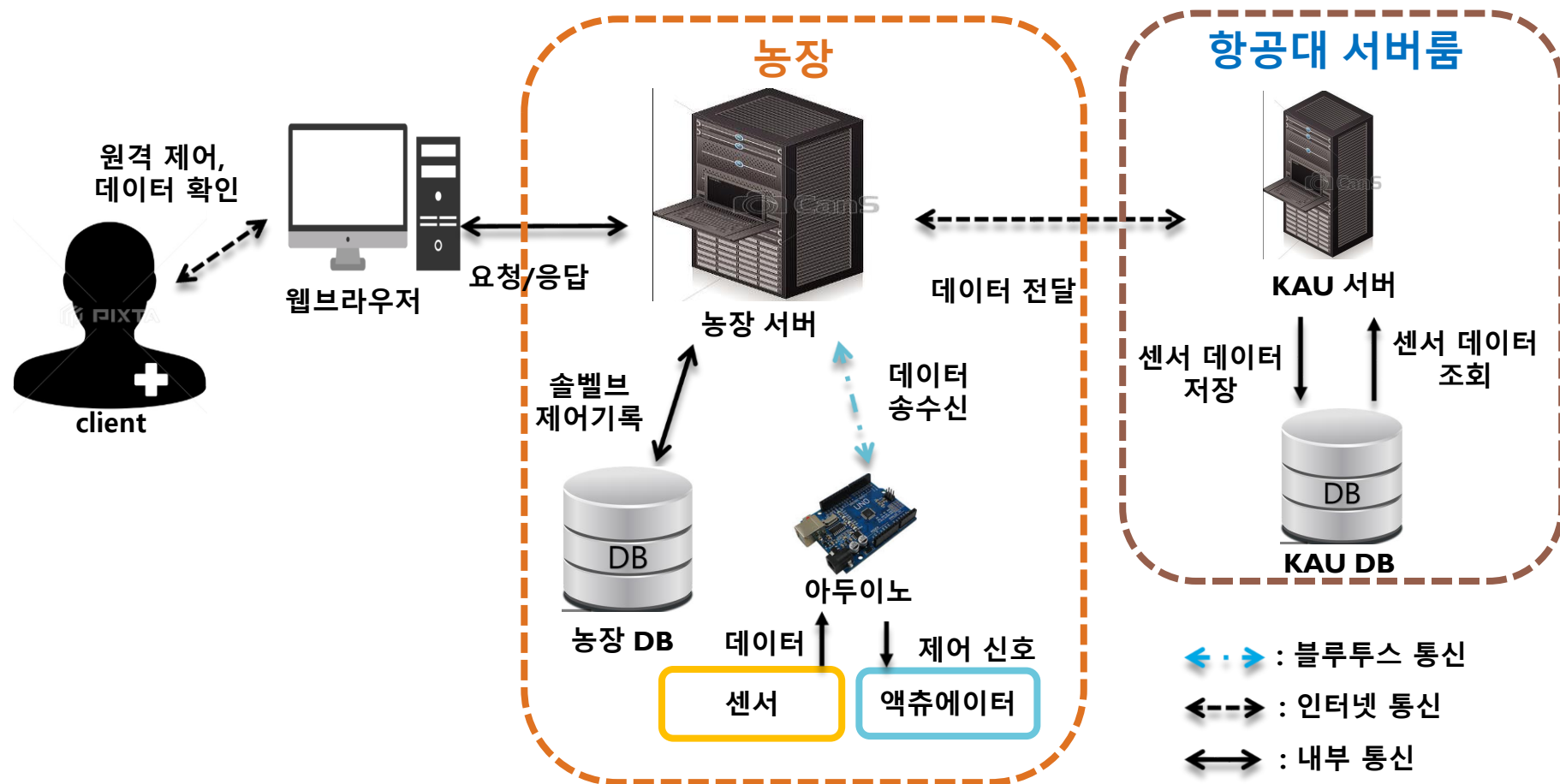
## ▶ IoT 스마트팜에 AI 자동관수제어 적용

- ▶ 시스템의 한계를 극복하기 위해 기존 스마트팜시스템으로 수집한 센서 데이터의 학습을 기반으로 자동제어를 접목시키자 함
  - ▶ DB에 축적한 센서 데이터들을 이용한 학습을 통해 시스템 스스로 적절한 판단으로 관수제어



# 연구진행사항(1/17)-Overview

## ▶ 본 연구실의 스마트팜 시스템의 구성

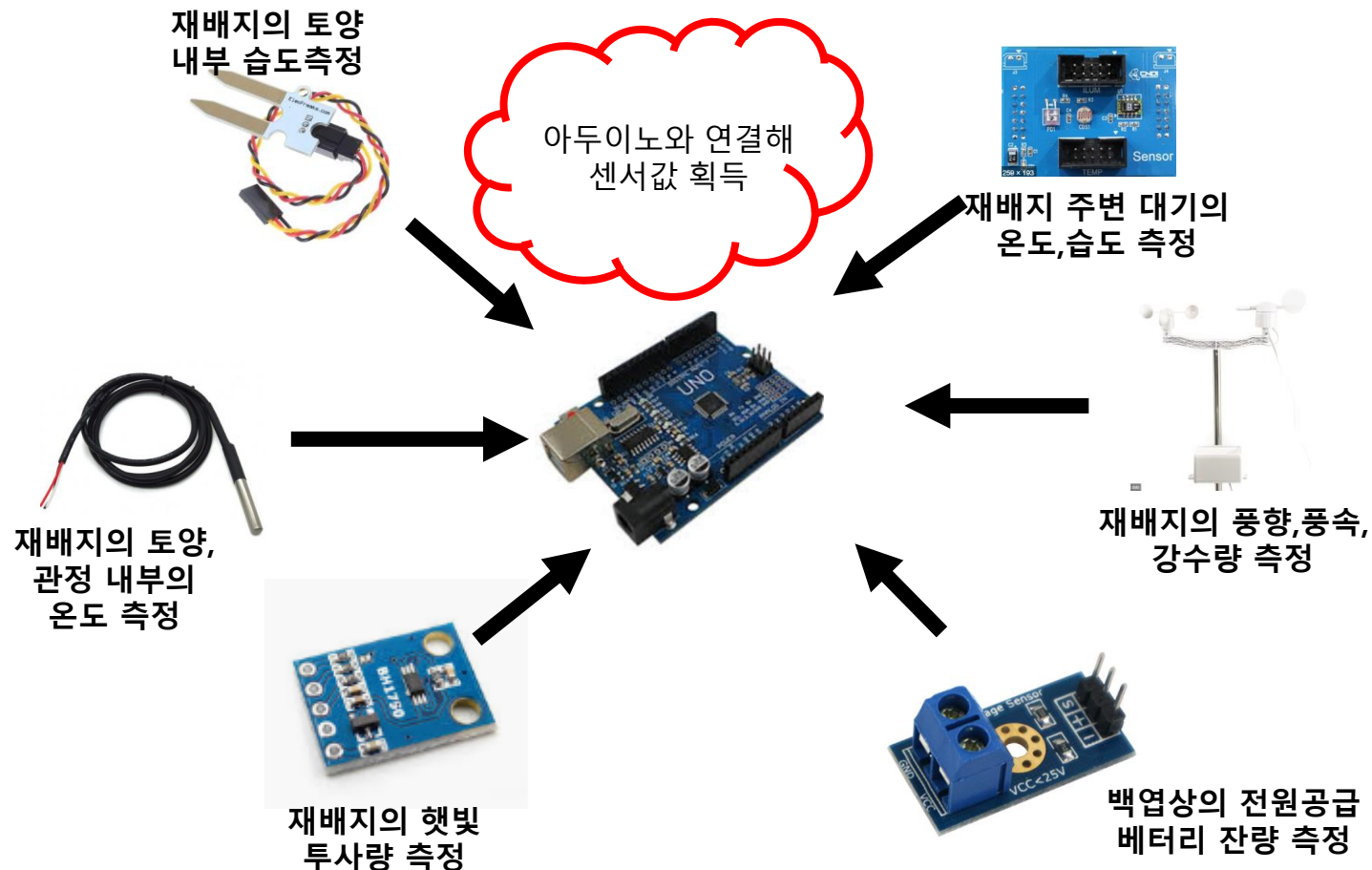


▶ 농장에서 작물재배에 영향을 끼치는 온도, 습도, 강수량 등을 센서가 전기 신호로 바꾸고 아두이노가 이를 받아 사람이 알 수 있는 수치로 바꿔줌



# 연구진행사항(3/17) - 스마트팜의 센서

## ▶ 스마트팜 시스템을 구성하는 센서들의 설치목적





# 연구진행사항(4/17) – 스마트팜의 센서

## ▶ 아두이노와 라즈베리파이의 데이터 송수신

### ▶ 블루투스의 사용

- ▶ 아두이노에 HC-06 블루투스 통신모듈 연결
  - TX,RX핀을 연결한 후 softwareSerial.h라이브러리를 이용
- ▶ 최초 연결시 라즈베리파이의 내장 블루투스 모듈과 각 아두이노의 블루투스 모듈을 명령어를 통해 페어링

```
pi@raspberrypi:~ $ bluetoothctl
[bluetooth]# power on
[bluetooth]# scan on
[bluetooth]# agent on
[bluetooth]# pair [MAC address]
[Bluetooth]# trust [MAC address]
[Bluetooth]# info [MAC address]
[bluetooth]# quit
```

<라즈베리파이 명령창에서의 페어링 과정을 위한 명령어>

### ▶ USB의 사용

- ▶ 아두이노를 라즈베리파이의 usb포트에 연결시 /dev 경로에 파일 생성
- ▶ 아두이노의 정보를 담고있는 파일로써 통신 경로로 사용가능

# 연구진행사항(5/17) – 스마트팜의 센서

## ▶ 센서의 제원 및 특징(1)

	아두이노와의 연결	측정범위 및 정격전압	특징
온/습도센서모듈 (CNDI)	Analog Pin x2	온도:-55°C ~ +150°C 습도: 0~100% 정격전압 5V	<ul style="list-style-type: none"><li>- SHT1X.h라이브러리를 이용해 온도값과 습도값 출력</li><li>- 모듈은 온도,습도센서와 A/D변환기, OTP메모리로 구성</li><li>- I2C통신 사용</li><li>- 트랜지스터의 온도에 따른 전류변화로 온도측정</li><li>- 습도에 따른 유전율 변화를 이용해 캐패시터로 습도 측정</li></ul>
토양 내부 습도센서 (Octopus soil moisture sensor)	Analog Pin x1	습도:0~100% 정격전압5V	<ul style="list-style-type: none"><li>- 토양이 수분 함유에 따른 미세한 저항값을 이용</li><li>- 센서에서 전류를 흘려 이에 따른 전압을 측정</li><li>- 아두이노 UNO의 10bit의 A/D 변환기로 0~1023값으로 측정된 전압값을 변환 후 출력</li></ul>
토양 내부온도센서 (DS18B20)	Digital Pin x1	-55 ~ +175°C 정격전압5V	<ul style="list-style-type: none"><li>- DallasTemperature.h , OneWire.h 라이브러리를 이용 하여 온도값 출력</li><li>- I2C 통신 사용</li><li>- 트랜지스터의 온도에 따른 전류변화로 온도측정</li></ul>

# 연구진행사항(6/17) – 스마트팜의 센서

## ▶ 센서의 제원 및 특징(2)

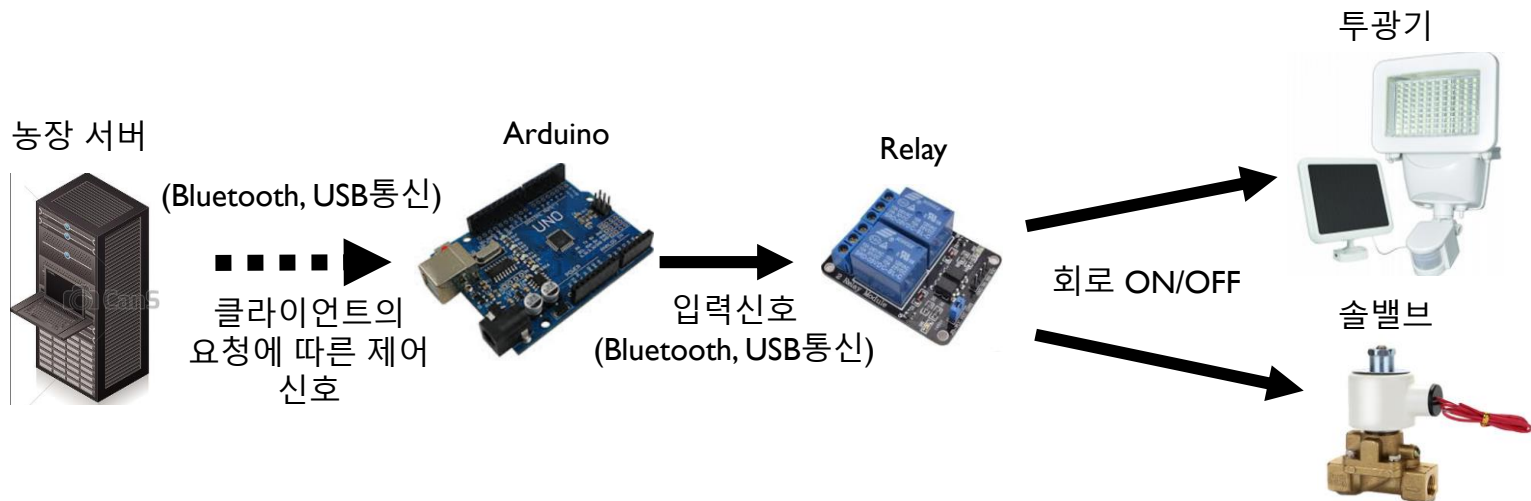
	아두이노와의 연결	측정범위 및 정격전압	특징
풍향/풍속/강수량 센서 (sparkfun weather meter)	Digital pin x2 Analog pin x1	풍향 : 16방향 표현 풍속, 강수량 : 측정제한없음 정격전압 : 5V	<ul style="list-style-type: none"><li>- 각기 다른 값을 가진 저항이 방위별로 연결되어 전류를 측정하여 풍향을 표현</li><li>- 풍속과 강수량은 수력과 풍력에 따라 내부의 스위치 가 스위칭 되는 횟수를 일정시간측정하여 연산</li></ul>
전압센서 (voltage sensor)	Analog pin x1	0~25V 정격전압 : 5V	<ul style="list-style-type: none"><li>- 내부회로의 저항들이 옴의법칙에 따라 0~25V의 전압 을 아두이노가 읽을 수 있는 0~5V로 변환</li></ul>
조도센서 모듈 (GY-302)	Analog pin x2	0~65335lux 정격전압 : 5V	<ul style="list-style-type: none"><li>- BH.1750h 라이브러리를 이용해 Lux 값 출력</li><li>- I2C통신 사용</li><li>- CDS센서, 16bit A/D변환기, 메모리로 구성</li><li>- 금속의 광전효과를 이용</li></ul>

# 연구진행사항(7/17) – 릴레이

## ▶ 농장에 구성된 기기의 원격제어를 위해 릴레이사용

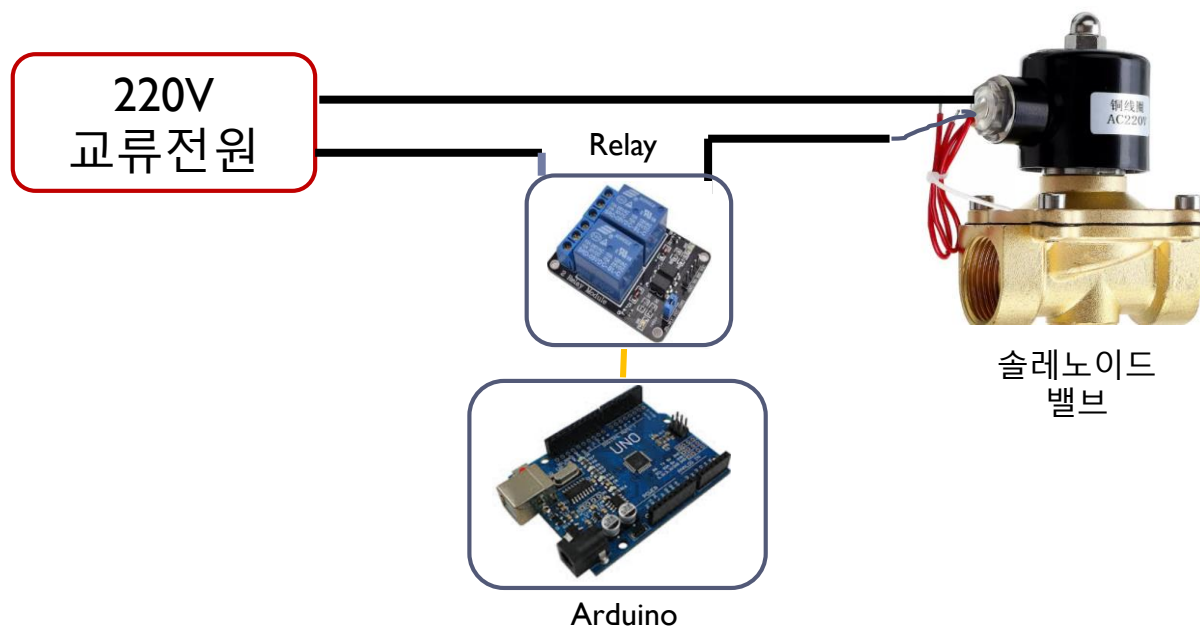
### ▶ 릴레이란?

- 입력신호에 따라 외부회로를 ON,OFF하는 스위치 (Binary actuator)
- 5V의 공급전원으로 최대전압 250V, 최대통전전류 10A의 회로에 사용가능



# 연구진행사항(8/17) – 솔레노이드 밸브

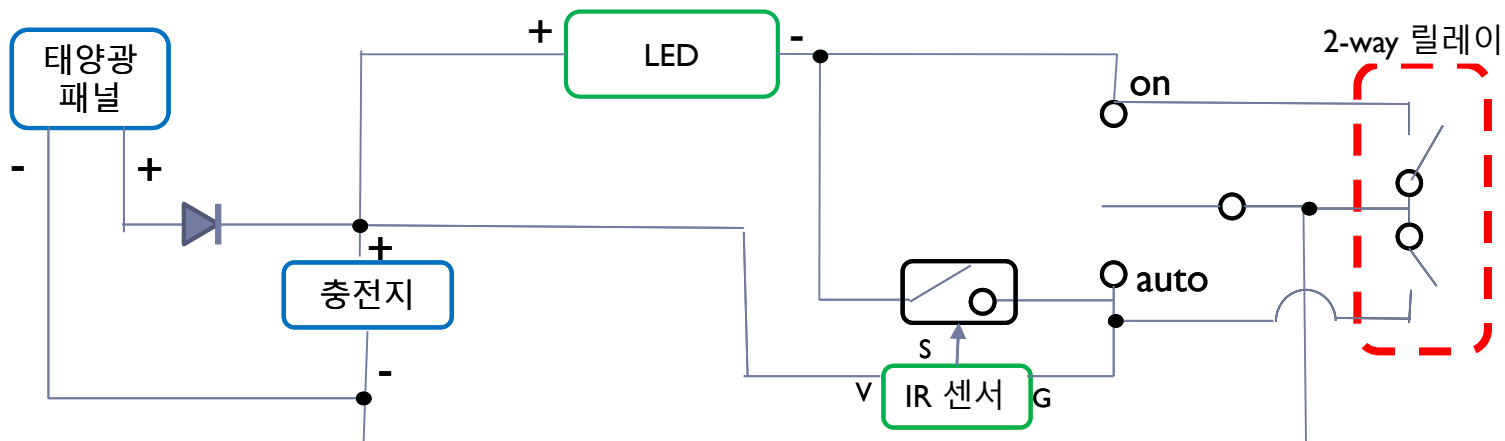
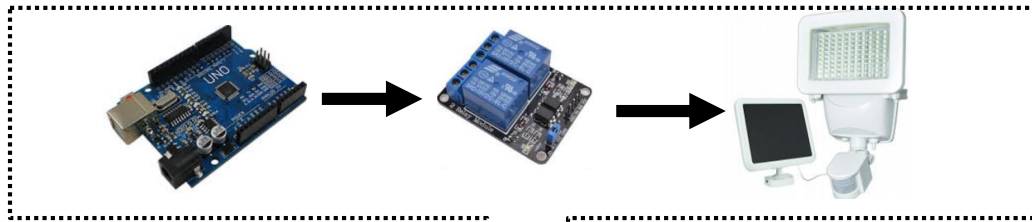
- ▶ 솔레노이드 밸브(HPW2140)
  - ▶ 사용목적: 농업용수로를 개방,폐쇄하여 물의 공급을 제어가능하게 함
  - ▶ 220V전압을 이용하며, 제어 신호를 보내 릴레이를 열고 닫음으로써 전원을 ON,OFF
  - ▶ HPW2140은 평시 닫힘 상태의 제품으로 전원이 인가될 때만 밸브를 개방

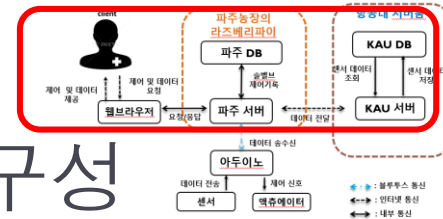


# 연구진행사항(9/17) – 투광기

## ▶ 투광기(Sunforce 100)

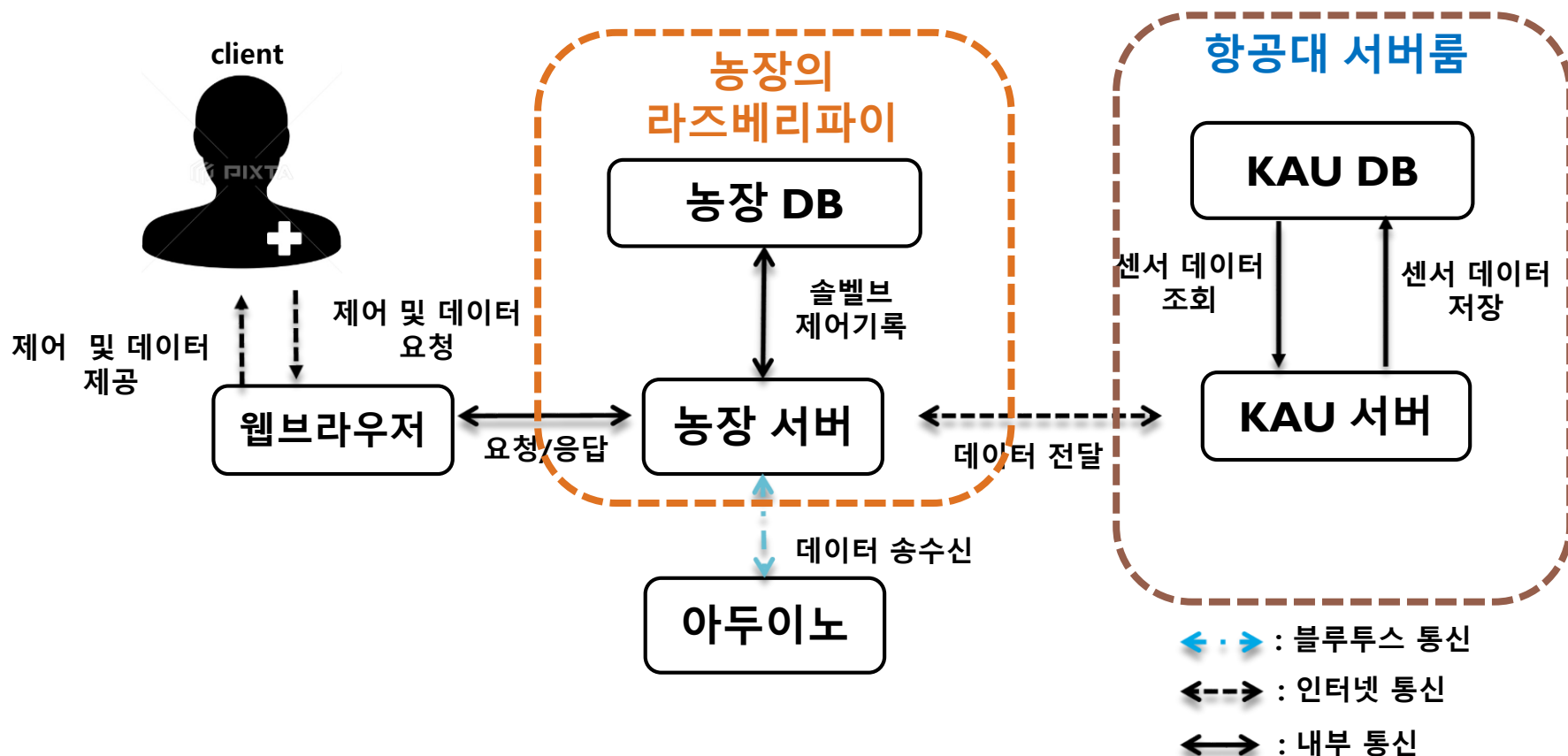
- ▶ ON,OFF,AUTO 세가지상태로 일정지역에 빛을 투사
- ▶ 태양광패널을 이용해 전원충전





# 연구진행사항(10/17)-스마트팜 서버구성

## ▶ Node.js를 이용한 웹/DB서버의 구성 및 통신



# 연구진행사항(11/17)-농장의 웹서버

## ▶ 데이터 확인, 원격제어를 위한 웹서버

### 스마트 팜 시스템

로그아웃(logout)

#### 중요 데이터 미리보기

현재 실외온도: -2.88 °C

현재 비닐하우스온도: -1.90°C / -1.86°C

현재 실외습도: 77.83%

현재 비닐하우스습도: 92.88% / 92.88%

강수량: 0.00mm

배터리 잔량: 0.00%

현재 관정온도: 2.69°C

#### 제어 (관수 / LED)

#### 기상관측

관수

LED

슬밸브 개폐를 통한 작물관수 제어

현재 상태(버섯용): 폐쇄

버섯용 개방

버섯용 폐쇄

현재 상태(잔디용): 폐쇄

잔디용 개방

잔디용 폐쇄

#### 농막의 LED제어

LED 켜기

LED 끄기

LED 자동

#### house의 LED제어

LED 켜기

LED 끄기

LED 자동

온도/습도/조도/강수량

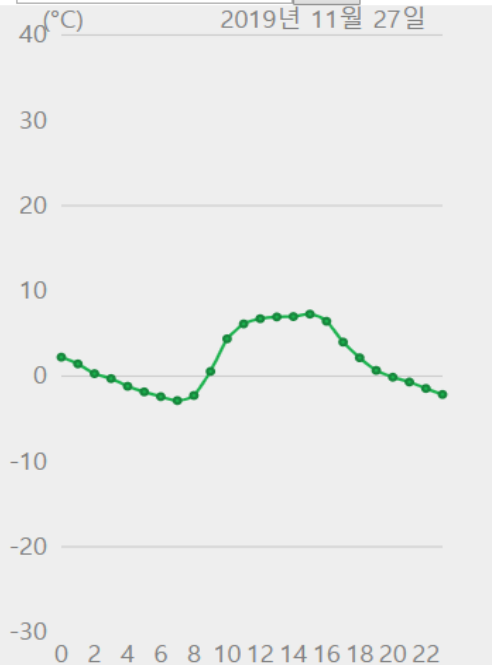
풍향/풍속

돌아가기

11/27/2019

검색

2019년 11월 27일



최저 온도: -2.86°C

최고 온도: 7.28°C

평균 온도: 1.71°C



# 연구진행사항(12/17)-농장의 웹서버

## ▶ 농장의 실시간 측정 데이터 표시

클라이언트  
스마트 팜 시스템

로그아웃 (logout)

### 중요 데이터 미리보기

현재 실외온도: -2.88 °C  
현재 비닐하우스온도: -1.90°C / -1.86°C  
현재 실외습도: 77.83%  
현재 비닐하우스습도: 92.88% / 92.88%  
강수량: 0.00mm  
배터리 잔량: 0.00%  
현재 관정온도: 2.69°C

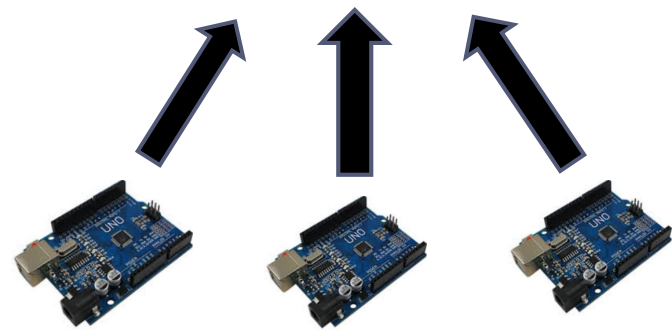
제어 (관수 / LED)

기상관측

실시간 센서값  
데이터 요청

실시간 데이터 정보제공

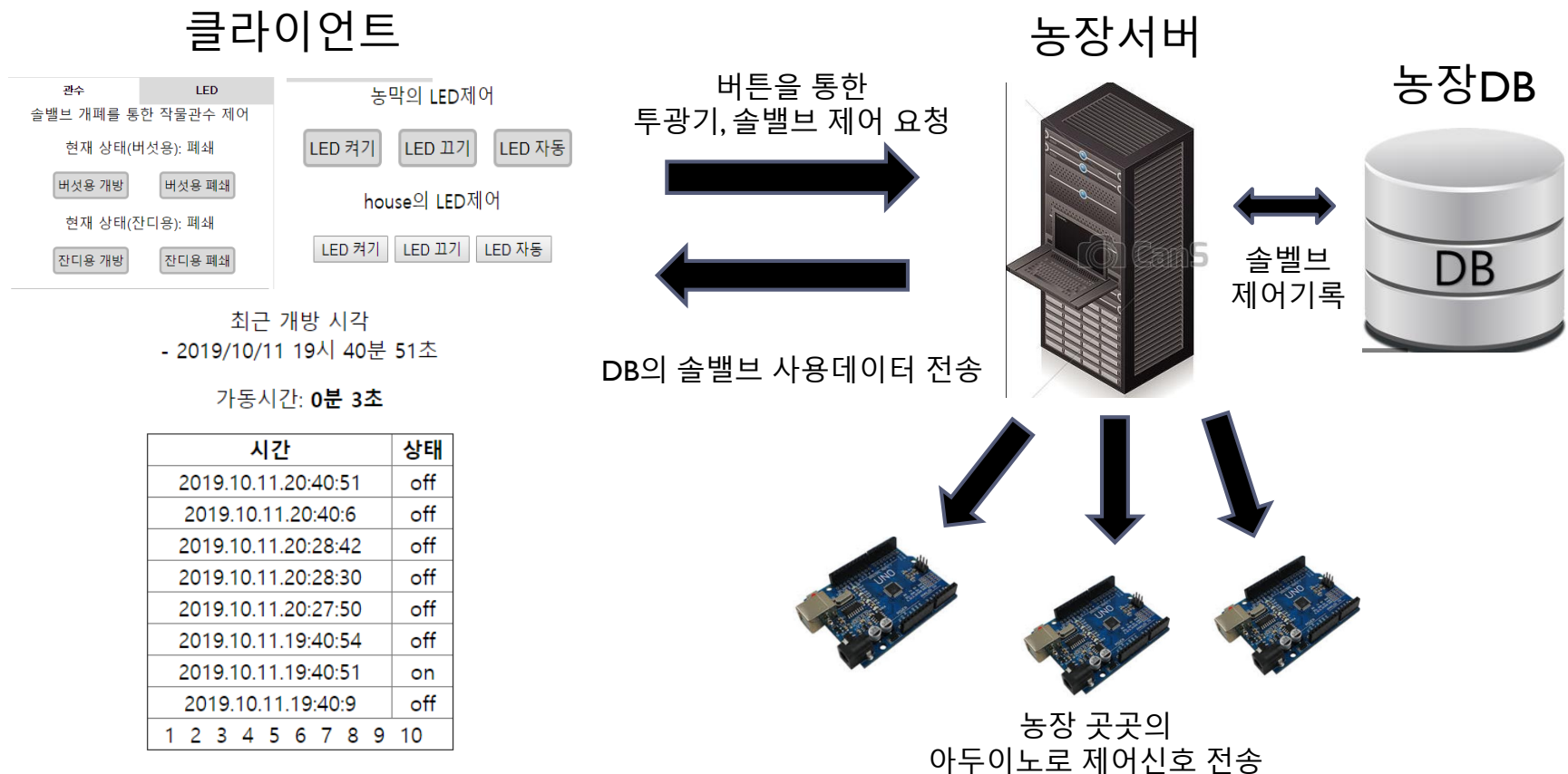
농장서버



농장 곳곳의  
아두이노로부터 센서값 수신

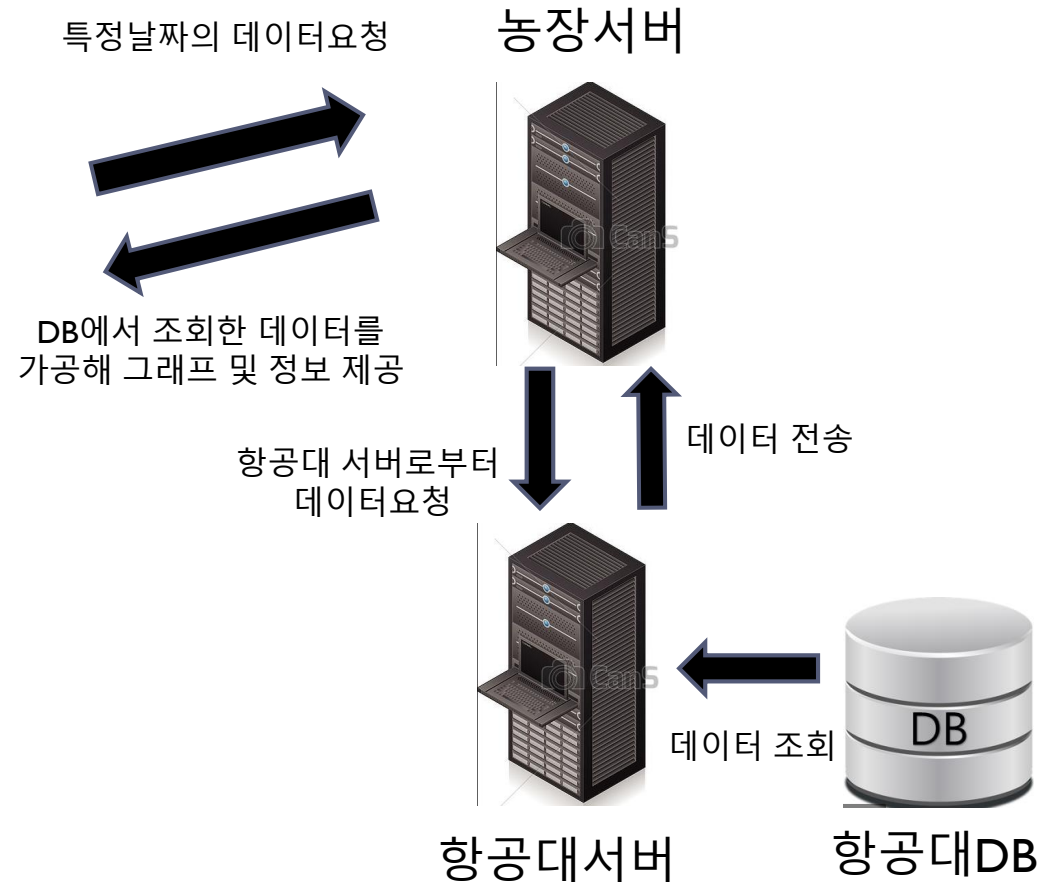
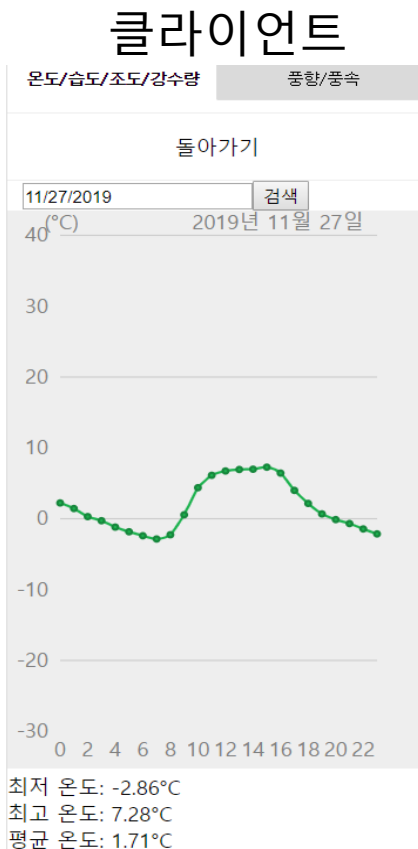
# 연구진행사항(13/17)-농장의 웹서버

## ▶ 제어 화면 및 가동기록 표시



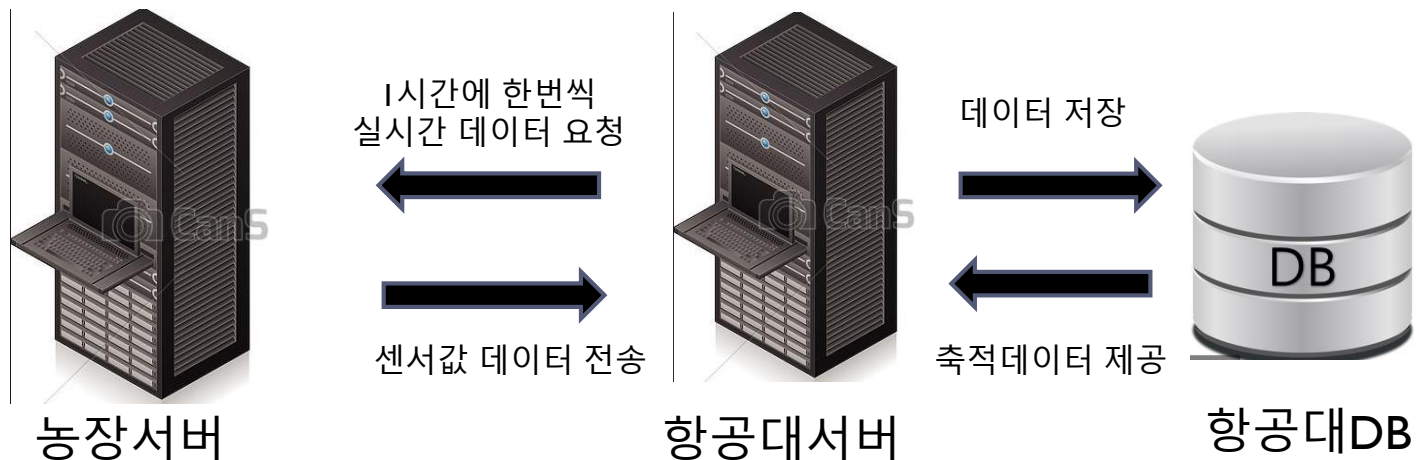
# 연구진행사항(14/17)-농장의 웹서버

## ▶ 그래프를 통한 시간별 상태변화 표시



# 연구진행사항(15/17)-항공대 DB서버

- ▶ 데이터 축적, 활용을 위한 항공대DB서버
  - ▶ 농장서버로 1시간에 한번씩 요청을 보내 실시간 데이터를 가져옴
  - ▶ DB에 센서값을 날짜, 시간별로 저장
  - ▶ 농장서버의 요청에 따라 DB에 저장한 데이터를 조회하여 전송



# 연구진행사항(16/17)-항공대 DB서버

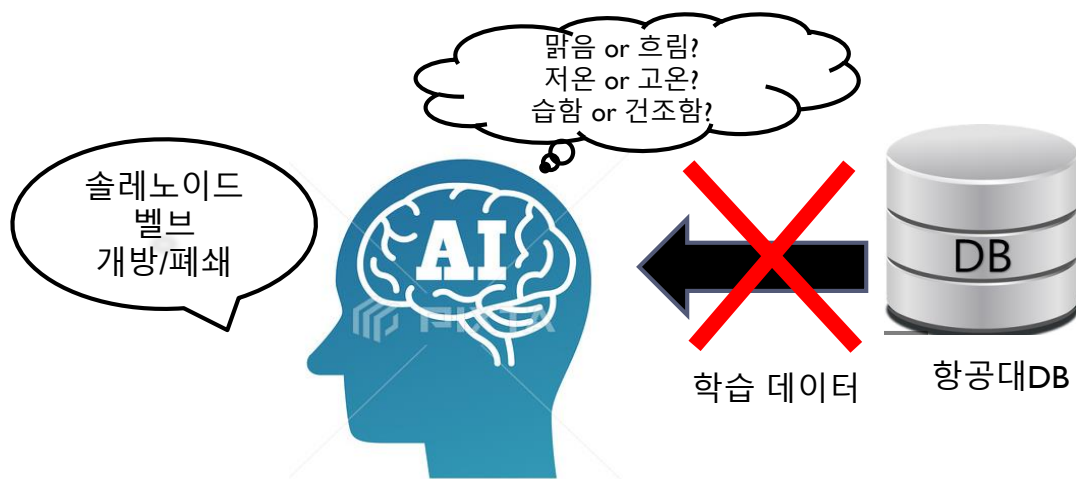
## ▶ 항공대 DB에 저장된 스마트 팜 센서 데이터

```
mysql> select *from weather where time like '2019-11-7%';
```

time	temperature	humidity	windSpeed	windName	rainGauge
2019-11-7-0	4.74	88.78	0.00	N	0.00
2019-11-7-1	4.44	89.09	0.00	SE	0.00
2019-11-7-2	4.29	90.16	0.00	NW	0.00
2019-11-7-3	4.46	90.74	0.00	NW	0.00
2019-11-7-4	4.12	90.61	0.00	W	0.00
2019-11-7-5	3.72	91.45	0.00	N	0.00
2019-11-7-6	3.16	92.14	0.00	ESE	0.00
2019-11-7-7	2.84	92.47	0.00	N	0.00
2019-11-7-8	3.54	92.86	0.00	N	0.00
2019-11-7-9	8.26	83.90	3.12	SW	0.00
2019-11-7-10	11.57	59.87	3.84	S	0.00
2019-11-7-11	12.96	51.92	0.00	E	0.00
2019-11-7-12	13.95	36.35	6.00	SSE	0.00
2019-11-7-13	14.38	31.69	9.60	S	0.00
2019-11-7-14	14.96	29.22	5.76	SSW	0.00
2019-11-7-15	14.10	29.43	1.68	S	0.00
2019-11-7-16	13.52	30.28	0.00	SE	0.00
2019-11-7-17	8.86	44.56	1.44	N	0.00
2019-11-7-18	6.41	56.57	0.00	N	0.00

# 연구진행사항(17/17)

- ▶ 자동관수제어 구현을 위해 AI 공부를 진행 중에  
**문제점 발견**
  - ▶ 센서 데이터는 DB에 지속적으로 수집중
  - ▶ 하지만 이를 AI에 대한 학습데이터로써 사용이 불가능
    - ▶ 센서 데이터에 따른 솔레노이드 밸브 제어 데이터가 없음



## 추후연구계획(1 / 1)

- ▶ AI 자동관수제어 구현에 적합한 데이터 분석, 수집
  - ▶ 지속적으로 축적되고 있는 데이터의 활용방안 연구
  - ▶ 목적에 맞는 추가적으로 필요한 데이터에 대한 조사와 그에 따른 데이터 획득을 위한 센서 추가설치

현재 수집 중인 데이터	대기 온/습도 , 토양 온/습도, 조도, 강수량, 풍향, 풍속.관정온도, 배터리전압 데이터
추가적으로 연구할 데이터	CO2농도, 토양산성/염류 데이터

---

▶ 감사합니다



# 첨부자료(주요 소스코드)

- ▶ 블루투스 통신을 위한 소스코드(서버측)
  - ▶ Exec함수를 이용한 블루투스 binding과 Serialport, fs모듈을 이용한 데이터 수신,송신

```
var SerialPort = require("serialport");
var fs = require('fs');
var exec = require('child_process').exec;
exec('sudo rfcomm bind /dev/rfcomm6 00:21:13:00:4F:0D 9 &')
exec('sudo rfcomm bind /dev/rfcomm3 98:D3:32:30:8A:F5 &')
```

⋮

```
var port = new SerialPort('/dev/rfcomm3', {
  baudRate: 9600
});
```

⋮

⋮

```
port.on('data', function(data){
  var decoder = new StringDecoder('utf8');
  var textData = decoder.write(data);
  line += textData
  var li = line.split("\n");
  if(li.length>1){
    sensor_value = li[li.length-2]
    line = ''
  }
})
```

<센서값 수신>

```
fs.open('/dev/rfcomm0', 'a', 666, function(e, fd){
  if(flag==0){
    fs.write(fd, '0', null, null, null, function(){
      fs.close(fd, function(){});
    });
  }
})
```

<제어값 송신>

# 첨부자료(주요 소스코드)

- ▶ 블루투스 통신을 위한 소스코드(아두이노측)
  - ▶ softwareSerial.h를 이용해 0,1번 핀이외의 핀으로 시리얼통신이 가능하며 각 핀에 블루투스모듈을 연결하여 이용

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(2, 3);
```

·  
·  
·

```
BTSerial.begin(9600);
```

·  
·  
·

```
BTSerial.print(str_tempC);
BTSerial.print(',');
BTSerial.print(str_humidity);
BTSerial.print(',');
BTSerial.print(str_tempC_1);
BTSerial.print(',');
BTSerial.print(str_humidity_1);
BTSerial.print('\n');
```

<센서값 송신>

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(2, 3);
```

·  
·  
·

```
BTSerial.begin(9600);
```

·  
·  
·

```
if (BTSerial.available() > 0 ) {
  char command = BTSerial.read();
  switch (command) {
    case '1':
      BTSerial.println("ON모드 활성화.\n");
      digitalWrite(8, LOW);
      digitalWrite(9, HIGH);

      break;
```

<제어값 수신>

# 첨부자료(주요 소스코드)

- ▶ USB를 이용한 데이터 송신
  - ▶ 아두이노를 라즈베리파이의 usb포트에 연결했을 때 라즈베리파이 측의 /dev 디렉토리안에 생성되는 파일이 아두이노와의 연결 경로가 됨
    - 예) 농장 IoT의 라즈베리파이에서의 경로 (/dev/ttyACM0)
  - ▶ 아두이노에서 소스를 업로드 할 때 툴 → 포트에서 ttyACM0을 선택

```
var fs = require('fs');
fs.open('/dev/ttyACM0', 'a', 666, function(e, fd) {
  if(flag==0){
    fs.write(fd, '0', null, null, null, function(){
      fs.close(fd, function({}));
    });
  }
}
```

# 첨부자료(주요 소스코드)

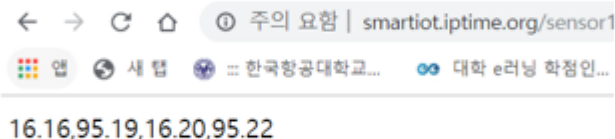
## ▶ 웹에서의 데이터 표현

- ▶ Port.on 메소드를 이용해 받은 sensor\_value를 사용자의 요청에 따라 응답으로 보내줌
- ▶ Sensor\_value 는 각 아두이노에 연결된 여러 센서값을 문자열로 저장



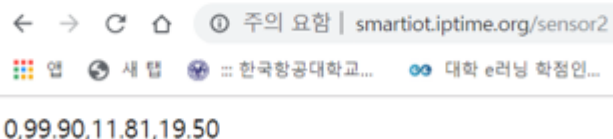
← → ↻ ⌂ ① 주의 요함 | smartiot.iptime.org/sensor  
앱 새 탭 :: 한국항공대학교... 대학 e러닝 학점인...  
16.40,91.69,0.00,NE,0.00

<실외온도, 실외습도, 풍속,풍향,강수량 데이터>



← → ↻ ⌂ ① 주의 요함 | smartiot.iptime.org/sensor1  
앱 새 탭 :: 한국항공대학교... 대학 e러닝 학점인...  
16.16,95.19,16.20,95.22

<비닐하우스내 온/습도 데이터>



← → ↻ ⌂ ① 주의 요함 | smartiot.iptime.org/sensor2  
앱 새 탭 :: 한국항공대학교... 대학 e러닝 학점인...  
0,99.90,11.81,19.50

<조도,토양내 습도,베터리전압, 토양내 온도 데이터>

```
app.get('/sensor', function(request, response){
  response.send(sensor_value);
});

app.get('/sensor1', function(request, response){
  response.send(sensor_value1);
});

app.get('/sensor2', function(request, response){
  response.send(sensor_value2);
});
```

<소스코드>

## 첨부자료(주요 소스코드)

- ▶ **센서값이 표시되는 과정(ex.중요데이터 미리보기의 현재 실외온도)**

## 중요 데이터 미리보기

현재 실외온도: 20.24 °C

## Led.ejs : 웹페이지에서의 표현

[illegible]

## test.js : 서버에게 데이터를 요청 및 처리

20.41,44.65,0.00,SW,0.00

```
$.get('/sensor2', function(data){
    var splice = data.split(',');
    $('#important_out_temp').append('<b>'+splice[0]+' &#176;C</b>');
    $('#important_out_hum').append('<b>'+splice[1]+'%</b>');
});
```

## 요청



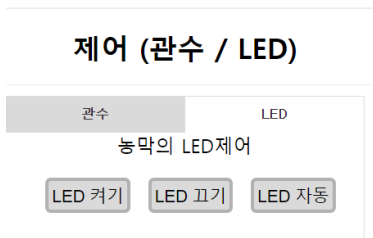
**답이요**

## app.js :요청에 따른 데이터를 응답으로 보냄

```
app.get('/sensor', function(request, response){
  response.send(sensor_value);
});
```

# 첨부자료(주요 소스코드)

- ▶ 웹을 통한 원격제어(Ex. 농막 투광기의 ON)



## Led.ejs : 웹페이지에 버튼을 표시

```
<p> 농막의 LED제어 </p>
<p><input type="button" id="onLED" class="ledButton" value="LED 켜기"></input>
<input type="button" id="offLED" class="ledButton" value="LED 끄기"></input>
<input type="button" id="autoLED" class="ledButton" value="LED 자동"></input></p>
```



## Buttoncontrol.js : 버튼클릭시의 동작

```
$('#onLED').click(function(){
    if(confirm("LED를 켜시겠습니까?")==true){
        $.post('solarstatus', {switchs:'on'}, function(){});
    }else{
        return;
    }
});
```

## 첨부자료(주요 소스코드)

### App.js : 요청에 따라 아두이노로 값을 전송

```
var switchs = request.body.switchs;
if(switchs=='off'){
    setSolar(0);
    response.redirect('/');
}
if(switchs=='on'){
    setSolar(1);
    response.redirect('/');
}
if(switchs=='auto'){
    setSolar(2);
    response.redirect('/');
}
});
```

```
app.post('/solarstatus', function(request, response){
    function setSolar(flag){
        var fs = require('fs');
        fs.open('/dev/rfcomm0', 'a', 666, function(e, fd){
            if(flag==0){
                fs.write(fd, '0', null, null, null, function(){
                    fs.close(fd, function(){});
                });
            }
            if(flag==1){
                fs.write(fd, '1', null, null, null, function(){
                    fs.close(fd, function(){});
                });
            }
            if(flag==2){
                fs.write(fd, '2', null, null, null, function(){
                    fs.close(fd, function(){});
                });
            }
        });
    }
});
```

### 아두이노의 투광기 제어부분 소스코드

```
case '1':
    BTSerial.println("ON모드 활성화.\n");
    digitalWrite(8, LOW);
    digitalWrite(9, HIGH);
```

# 첨부자료(주요 소스코드)

- ▶ 웹을 통한 원격 제어(Ex.솔레노이드 밸브 개방)

현재 상태(버섯용): 폐쇄

버섯용 개방

버섯용 폐쇄



## Led.ejs : 웹페이지에 버튼 표현

```
<p>
<input type="button" id="turnOnButton" class="solbutton" value="버섯용 개방"></input>
<input type="button" id="turnOffButton" class="solbutton" value="버섯용 폐쇄"></input>
</p>
```



## Buttoncontrol.js : 버튼 클릭시의 동작

```
$('#turnOnButton').click(function () {
    if(confirm("버섯용 솔레노이드 밸브를 개방하시겠습니까?")==true){
        $.post('/onoff', {switchs:'on'}, function() {});
    }
});
```



# 첨부자료(주요 소스코드)

- ▶ 웹을 통한 원격제어(Ex. 솔벨브의 ON)

## App.js : 아두이노에 값을 전송

```
app.post('/onoff', function(request, response) {  
  function setVALVE(flag){  
    var fs = require('fs');  
    fs.open('/dev/ttyACM0', 'a', 666, function(e, fd) {  
      if(flag==0){  
        fs.write(fd, '0', null, null, null, function(){  
          fs.close(fd, function(){});  
        });  
      }  
      if(flag==1){  
        fs.write(fd, '1', null, null, null, function(){  
          fs.close(fd, function(){});  
        });  
      }  
      if(flag==2){  
        fs.write(fd, '2', null, null, null, function(){  
          fs.close(fd, function(){});  
        });  
      }  
      if(flag==3){  
        fs.write(fd, '3', null, null, null, function(){  
          fs.close(fd, function(){});  
        });  
      }  
    })  
  }  
})
```

# 첨부자료(주요 소스코드)

- ▶ 웹을 통한 원격제어(Ex. 솔벨브의 ON)

**App.js : 벨브를키는 시간을 농장서버DB에 저장하고, 1시간뒤 자동 폐쇄**

```
var switchs = request.body.switchs;

var d = new Date()
var time = d.getFullYear() + '.'+(d.getMonth()+1)+'.'+d.getDate()+ '.'+d.getHours()+ ':'+d.getMinutes()+ ':'+d.getSeconds();

var st_time;
if(switchs=='on'){
    pool.getConnection(function(err, connection){
        connection.query('insert into on_off_time (time, status) values("' +time+'", "' +switchs+'");', function(err, data){
            connection.release();
        })
    })
    setVALVE(0);
    st_time = setTimeout(function(){
        pool.getConnection(function(err, connection){
            var stop_time = new Date()
            var time2 = stop_time.getFullYear()+ '.'+(stop_time.getMonth()+1)+'.'+stop_time.getDate()+ '.'+stop_time.getHours()+ ':'+stop_time.getMinutes()+ ':'+stop_time.getSeconds();
            connection.query('insert into on_off_time (time, status) values("' +time2+'", "off");', function(err, data){
                connection.release();
            })
        });
        setVALVE(1);
    }, 3600000)
    response.redirect('/led');
```

# 첨부자료(주요 소스코드)

- ▶ 솔레노이드 밸브 사용에 따른 데이터의 표시

최근 개방 시각  
- 2019/10/11 19시 40분 51초

가동시간: 0분 3초

시간	상태
2019.10.11.20:40:51	off
2019.10.11.20:40:6	off
2019.10.11.20:28:42	off
2019.10.11.20:28:30	off
2019.10.11.20:27:50	off
2019.10.11.19:40:54	off
2019.10.11.19:40:51	on
2019.10.11.19:40:9	off
1 2 3 4 5 6 7 8 9 10	

## Led.ejs : 웹페이지에 데이터를 표시

```
<div id="look_record"><h4>버섯용 솔밸브 기록 보기</h4></div>
<div id="record_data" style="text-align:center;">
  <p id = "sol_on_time"></p>
  <p id = "on_off_time_at"></p>
  <div id="record" style="text-align:center;padding-left:30px;"></div>
</div>
```

# 첨부자료(주요 소스코드)

- ▶ 솔레노이드 밸브 사용에 따른 데이터의 표시

## buttoncontrol.js :

서버에 요청을 보내 데이터를 가져와 배열에 저장

```
$.getJSON('/record1', function (data) {  
    var time = new Array();  
    var status = new Array();  
    $(data).each(function (index, item) {  
        time.push(item.time);  
        status.push(item.status);  
        count++;  
    })  
    for(var i=0; i<count; i++) {  
        if(status[i] == "on") {  
            break;  
        } else {  
            recent_on_index++;  
        }  
    }  
})
```

## App.js :

DB의 테이블에서 조회한 데이터를 응답으로 전송

```
app.get('/record1', function(request, response){  
    pool.getConnection(function(err, connection){  
        connection.query('SELECT * from on_off_time1 order by id desc',  
            function(err, data){  
                response.send(data);  
                connection.release();  
            })  
        })  
    })
```

# 첨부자료(주요 소스코드)

- ▶ 솔레노이드 밸브 사용에 따른 데이터의 표시

## Buttoncontrol.js : Led.ejs의 각 Id의 태그에 데이터를 표시

```
$('#on_off_time_at').empty();
$('#on_off_time_at').append('가동시간: <b>'+minute+'분 '+second+'초</b>');
to=setTimeout(realTime, 1000);
})();
$('#sol_on_time').empty();
$('#sol_on_time').append("최근 개방 시각<br>- "+rot[0]+"년 "+rot[1]+"월 "+rot[2]+"일 "
    +rotr[0]+"시 "+rotr[1]+"분 "+rotr[2]+"초");
if(status[0]=='on'){
    $('#sol_status').empty();
    $('#sol_status').append('현재 상태(버섯용): 개방');
}else if(status[0]=='off'){
    $('#sol_status').empty();
    $('#sol_status').append('현재 상태(버섯용): 폐쇄');
}
if(count<8){
    for(var i=0; i<count; i++){
        output += '<tr><td>'+time[i]+'</td><td>'+status[i]+'</td></tr>'
    }
    for(var i=count; i<8; i++){
        output += '<tr><td>&nbsp;</td><td>&nbsp;</td></tr>'
    }
}else{
    for(var i=0; i<8; i++){
        output += '<tr><td>'+time[i]+'</td><td>'+status[i]+'</td></tr>'
    }
}
```

## 첨부자료(주요 소스코드)

- ▶ 항공대서버에서 DB로 데이터를 저장하는 과정
  - ▶ Mysql 모듈을 이용해 DB사용을 위한 객체생성

```
var pool = mysql.createPool({  
  host      : 'localhost',  
  user      : 'root',  
  password  : 'logiclab',  
  database  : 'weather'  
});
```

- ▶ DB에 테이블이 있는지 확인 후 없다면 생성

```
pool.getConnection(function(err, connection) {  
  connection.query('show tables like "weather";', function(err, data) {  
    if(data.length!=0){  
      isTable = 'yes'  
    }else{  
      isTable = 'no'  
    }  
    if(isTable=='no'){  
      connection.query('CREATE TABLE weather(time varchar(200) null, temperature var  
        if(error) throw error;  
      });  
    }else if(isTable=='yes'){  
      console.log('MySql have this table')  
    }  
    connection.release();  
  });  
});
```

※항공대 서버측 소스코드

# 첨부자료(주요 소스코드)

- ▶ GetHTML함수: 농장서버에 요청을 보내 받아온 센서값을 테이블에 저장하는 기능

← → ↻ 🏠 ① 주의 요함 | smartiot.iptime.org/sensor  
🌐 📄 🌐 :: 한국항공대학교... 🌐 대학 e러닝 학정인...  
16.40,91.69,0.00,NE,0.00

← → ↻ 🏠 ① 주의 요함 | smartiot.iptime.org/sensor1  
🌐 📄 🌐 :: 한국항공대학교... 🌐 대학 e러닝 학정인...  
16.16,95.19,16.20,95.22

← → ↻ 🏠 ① 주의 요함 | smartiot.iptime.org/sensor2  
🌐 📄 🌐 :: 한국항공대학교... 🌐 대학 e러닝 학정인...  
0.99.90,11.81,19.50

※각 경로에 요청을 보내면  
실시간 센서값을 응답으로 보내준다

```
function getHTML_1 () {  
    var d = new Date()  
    request.get({url: 'http://smartiot.iptime.org/sensor'}, function(err, response, html){  
        console.log(html);  
        if(html==null || html==undefined || html==" " || html==" "){  
            console.log('Page Not Found! Recrawling!');  
            getHTML_1();  
        }else{  
            var weather;  
            html = html.replace(/\r/g, '');  
            weather = d.getFullYear()+'-'+(d.getMonth()+1)+'-'+d.getDate()+'-'+d.getHours()+','+html;  
            weather = weather.split(',');  
            html = d.getHours()+':00,'+html;  
            html += '\n';  
        }  
    })  
}
```

※항공대 서버측 소스코드

# 첨부자료(주요 소스코드)

## ▶ GetHTML함수 소스코드 뒷부분

```
fs.appendFile(file_1, html, 'utf8', function(err){
    if(err) throw err;
    console.log('file write_1');
})
pool.getConnection(function(err, connection) {
    connection.query('Insert into weather (time, temperature, humidity, windSpeed, windName, rainGauge) values(?,?,?,
    ,[weather[0], weather[1], weather[2],weather[3],weather[4],weather[5]
    ], function() {
        connection.release();
    });
});
}
})
}
```

※항공대 서버측 소스코드



## 첨부자료(주요 소스코드)

- ▶ 1시간에 한번씩 getHTML함수가 실행되며 파일에도 값을 저장 (CSV 형식)

```
setInterval(function(){
    var d = new Date()
    if(day!=d.getDate()){
        console.log('in_1');
        file_1 = 'weather/'+d.getFullYear()+'-'+(d.getMonth()+1)+'-'+d.getDate()+' .csv';
        fs.open(file_1, 'w', function(err){
            if(err) throw err;
            console.log('file change_1');
        })
        getHTML_1();
        day = d.getDate();
    }else{
        getHTML_1();
    }
}, 3600000)
```

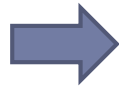
※항공대 서버측 소스코드

## 첨부자료(주요 소스코드)

- ▶ 항공대 DB에 저장한 데이터를 이용하는 과정
  - ▶ 농장서버에서 필요한 용도에 따라 요청을 보냄
    - ▶ 예1)오늘의 실외온도

기상관측	
온도/습도/조도/강수량	풍향/풍속
돌아가기	
현재 실외 / 관정 / 토양 온도	
오늘 실외 / 관정 / 토양 온도	
특정날짜 실외 온도	
특정날짜 관정 온도	
특정날짜 토양 온도	

click



```
});  
$('body').on('click', '#today_temp', function(){  
    $('#menu').empty();  
    ...  
    $.getJSON("http://210.119.30.212:8080/today_weather", function(data){  
        var wdt = [];  
        $.each(data, function(index, item){  
            time = item.time;  
            time = time.split('-');  
            time = time[3];  
            wdt.push({  
                time: parseInt(time),  
                temperature: parseFloat(item.temperature)  
            })  
            if(parseFloat(item.temperature)<low_temp){  
                low_temp=parseFloat(item.temperature);  
            }  
            if(parseFloat(item.temperature)>high_temp){  
                high_temp = parseFloat(item.temperature);  
            }  
            aver_temp+=parseFloat(item.temperature);  
        });  
        aver_temp = aver_temp / wdt.length;  
        aver_temp = aver_temp.toFixed(2);  
    });  
});
```

※농장 서버측 소스코드

## 첨부자료(주요 소스코드)

- ▶ 요청에 따라 Date객체로 오늘 날짜값을 구하여 테이블에서 오늘 저장된 데이터를 조회 후 응답으로 보내줌

```
app.get('/today_weather',function(request,response){
  var d = new Date()
  var today_checker='';
  today_checker=d.getFullYear()+'-'+(d.getMonth()+1)+'-'+d.getDate();
  pool.getConnection(function(err, connection) {

    connection.query("SELECT * FROM weather WHERE time like '"+today_checker+"'", function(err, data) {
      response.send(data);
      connection.release();
    });
  });
});
```

※항공대 서버측 소스코드

## 첨부자료(주요 소스코드)

---

- ▶ 예2)특정날짜 실외습도
  - ▶ 요청이 들어오면 테이블 내의 모든 데이터를 농장서버로 보내줌

```
app.get('/whole_weather',function(request,response){
  pool.getConnection(function(err, connection) {
    connection.query("SELECT * FROM weather;", function(err, data) {
      if (err)
        response.send(data);
      connection.release();
    });
  });
});
```

※항공대 서버측 소스코드

## 첨부자료(주요 소스코드)

- ▶ 항공대 서버로부터 응답으로 받은 데이터를 datepicker 객체를 이용해 선택한 날짜의 데이터만 분류해냄

```
$(function(){  
    $('#datepicker').datepicker({minDate: new Date(2016, 9,6), maxDate: 0});  
})
```

```
$.each(data, function(index, item){  
    time1 = item.time;  
    time1 = time1.split('-');  
    if(parseInt(time1[0])==parseInt(select_date[2])&&parseInt(time1[1])==parseInt(select_date[0]))  
        time1=time1[3];  
    wdt.push({  
        time: parseInt(time1),  
        temperature: parseFloat(item.temperature)  
    })  
    if(parseFloat(item.temperature)<low_temp){  
        low_temp=parseFloat(item.temperature);  
    }  
    if(parseFloat(item.temperature)>high_temp){  
        high_temp = parseFloat(item.temperature);  
    }  
    aver_temp+=parseFloat(item.temperature);  
});  
aver_temp = aver_temp / wdt.length;  
aver_temp = aver_temp.toFixed(2)
```

※농장 서버측 소스코드

## 첨부자료(주요 소스코드)

---

```
import tensorflow as tf
tf.set_random_seed(777)
import numpy as np
import math
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler()
xy=np.loadtxt('C:\\pajuDB\\trainDB5.csv',delimiter=',',dtype=np.float32)
x_data_train=xy[:200,0:-1]
x_data_test=xy[201:384,0:-1]
ex=[0.,0.,0.,0.,0.,0.]
ex_Y=[[1]]
y_data_train=xy[:200,-1]
y_data_test=xy[201:384,-1]

X=tf.placeholder(tf.float32, shape=[None,6],name='X-input')
Y=tf.placeholder(tf.float32, shape=[None,1],name='Y-input')
W1=tf.Variable(tf.random_normal([6,7]), name='weight1')
W2=tf.Variable(tf.random_normal([7,1]), name='weight2')
b1=tf.Variable(tf.random_normal([1,7]), name='bias1')
b2=tf.Variable(tf.random_normal([1]), name='bias2')

h1=tf.matmul(X,W1)+b1
hypothesis1=tf.sigmoid(h1)
h2=tf.matmul(hypothesis1,W2)+b2
hypothesis = tf.nn.sigmoid_cross_entropy_with_logits(logits=h2, labels=Y)
cost = tf.reduce_mean(hypothesis)
```

## 첨부자료(주요 소스코드)

---

```
train=tf.train.GradientDescentOptimizer(learning_rate=0.0001).minimize(cost)
predicted=tf.cast(hypothesis>0.5,dtype=tf.float32)
accuracy=tf.reduce_mean(tf.cast(tf.equal(predicted,Y),dtype=tf.float32))
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    feed = {X:x_data_train,Y:y_data_train}
    for step in range(10000):
        _,cost_val=sess.run([train,cost],feed_dict=feed)
        if step % 1000==0:
            print(step,cost_val)
    h,c,a=sess.run([hypothesis,predicted,accuracy],feed_dict={X:x_data_test,Y:y_data_test})
    print("\nhypothesis:",h,"\ncorrect Y:",c,"\naccuracy:",a)
```

# 연구진행사항(17/17)- ML기반 관수제어

---

## ▶ 학습을 위한 데이터 가공

### ▶ 시간별로 축적되고 있는 raw data를 학습데이터로 이용하고자 데이터를 가공

#### ▶ 가공기준

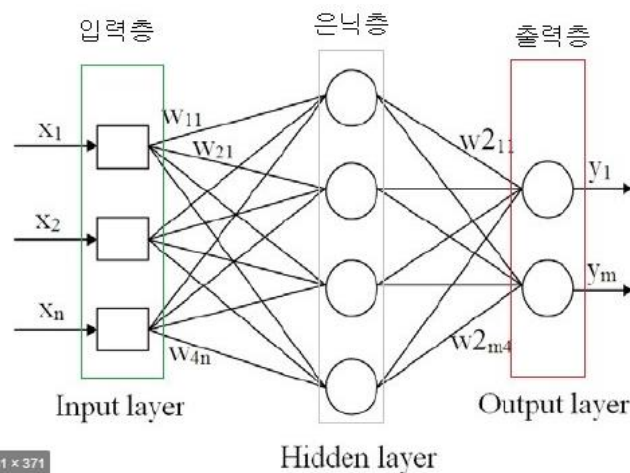
- 작물 성장에 영향을 직접적영향을 끼치는 데이터들을 선정 (조도, 토양내부의 온/습도, 외부 온/습도, 강수량)
- 농장농장의 재배품목인 고추에 대한 최적의 재배환경기준으로 솔밸브 ON/OFF (1과 0)를 판단 (기준 출처: 농사로,청양고추연구센터)
  - 예 1) 일정 토양온도 이상이며,강수가 없을 경우 솔밸브 개방,
  - 예 2)솔밸브를 개방하는 일정습도 이하지만, 조도가 0(밤)일 경우 솔밸브 차단
- 농사기간이 아닌 11월 중순~2월중순의 데이터 제외



# 연구진행사항(18/17)- ML기반 관수제어

## ▶ ML기반 관수제어의 구현

- ▶ Tensorflow를 이용한 다층 퍼셉트론(MLP)분류 모델 구성
  - ▶ 약 300개의 train data, 100개의 test data를 이용한 학습
  - ▶ Threshold 값을 기준으로 1과 0(솔벨브의 ON/OFF)을 분류
  - ▶ 학습결과 약 70%의 정답률을 보임
    - 적은 학습 데이터수, 데이터값의 편향으로 모델의 개선필요



# 연구진행사항(19/17)- ML기반 관수제어

## ▶ 다층퍼셉트론의 학습결과

```
0 0.6795847
1000 0.6522481
2000 0.627543
3000 0.60522735
4000 0.5850746
5000 0.56687593
6000 0.5504402
7000 0.53559095
8000 0.52216965
9000 0.51003236
10000 0.49904954
```

※1만회의 학습중 1000회마다  
감소하는 비용함수값 확인

```
hypothesis: [[0.7751738 ]
[0.7372587 ]
[0.5720087 ]
[0.7806734 ]
[0.7543092 ]
[0.8225873 ]
[0.9704495 ]
[0.7328505 ]
[0.6082699 ]
[0.8154201 ]
[1.1024282 ]
[0.3418597 ]
[0.26211712]
[0.53250112]]
```

※각데이터에 따른 정답예측  
(threshold=0.5 기준 분류모델)

```
correct Y: [[1.]
[1.]
[0.]
[1.]
[1.]
[1.]
[1.]
[1.]
[0.]
[1.]
[1.]
[1.]
[1.]
[1.]
```

※훈련데이터의 정답



accuracy: 0.7013889

---



---

---

---

---



---



---



## 제어 (관수 / LED)

관수

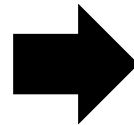
LED

농막의 LED제어

LED 켜기

LED 끄기

LED 자동



관수

LED

농막의 LED제어

LED 켜기

LED 끄기

LED 자동

house의 LED제어

LED 켜기

LED 끄기

LED 자동

## 스마트 팜 시스템

로그아웃(logout)

### 중요 데이터 미리보기

현재 실외온도: **-2.88 °C**

현재 비닐하우스온도: **-1.90°C / -1.86°C**

현재 실외습도: **77.83%**

현재 비닐하우스습도: **92.88% / 92.88%**

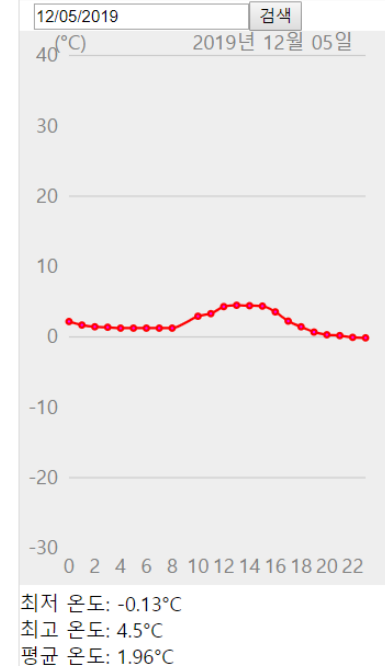
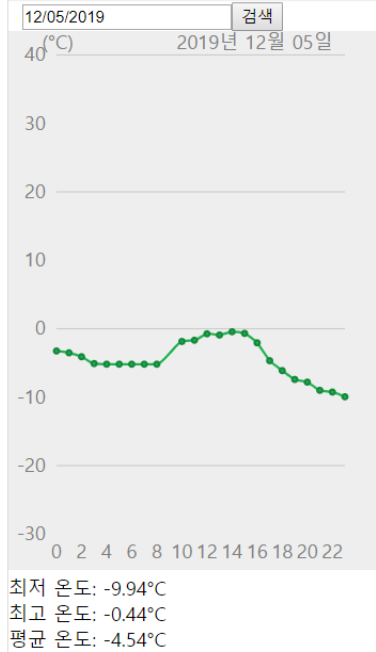
강수량: **0.00mm**

배터리 잔량: **0.00%**

현재 관정온도: **2.69°C**

제어 (관수 / LED)

기상관측



---

## Match making



## Game

