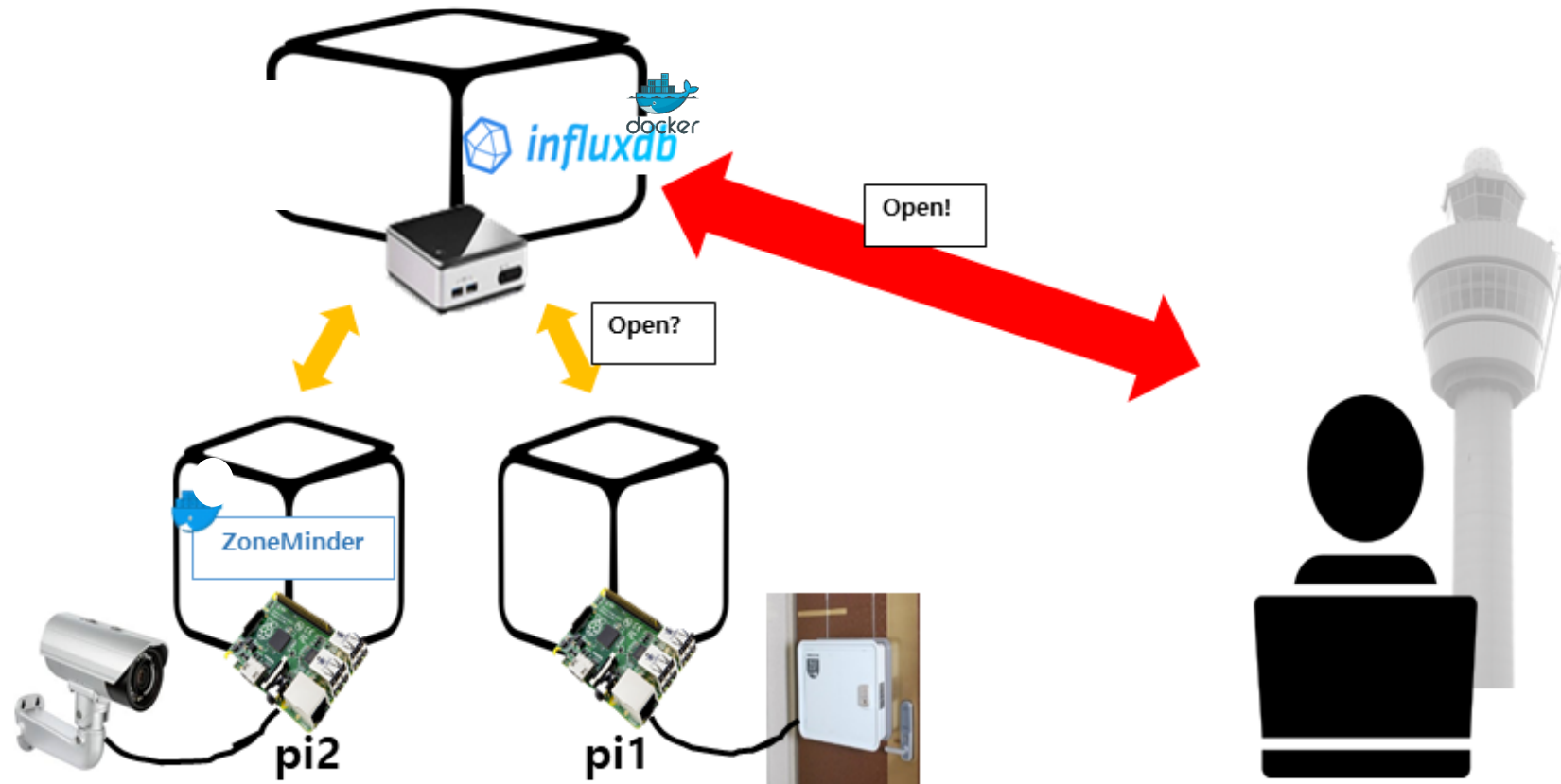




무인 택배함 관리 시스템

전기전자컴퓨터공학부
김은지

Overview



Software

ZoneMinder



소프트웨어 구성 요소

Docker

컨테이너 구동

InfluxDB

택배함 사용 현황 저장

Zoneminder

CCTV 실시간 모니터링

Hardware



사용된 하드웨어

라즈베리파이 2대

NUC 1대

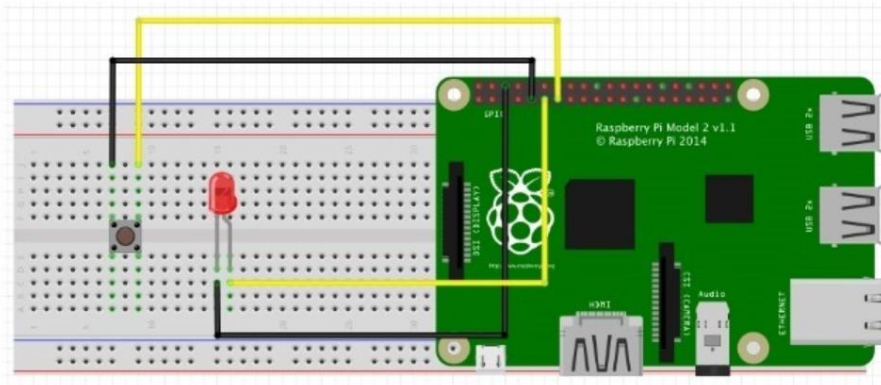
노트북 1대

유선 공유기 1대

스위치 허브 1대

웹캠 1대

Hardware



사용된 하드웨어

스위치 1개

Led 1개

브레드보드 1개

Working plan

11월 2주차

NUC & pi 작업 환경 세팅

11월 3주차

Pi 무인 택배함 구현, 택배함 기록 DB에 업로드

11월 4주차

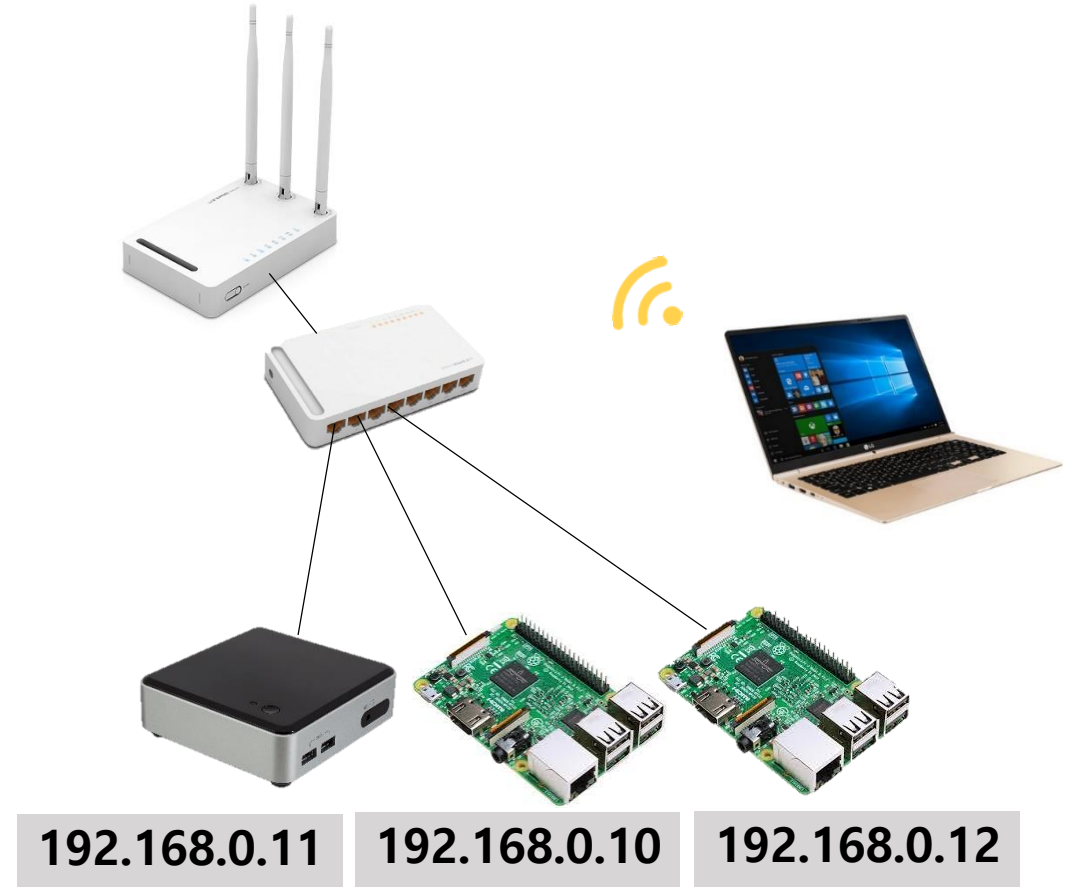
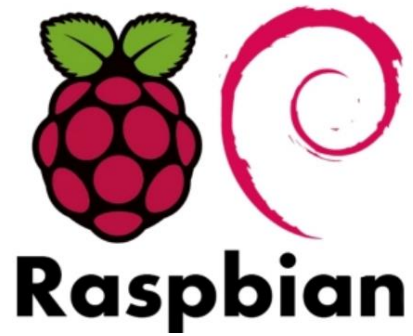
CCTV 구현

12월 1주차

수정 및 보완

Progress details

작업환경 세팅



Progress details

CCTV



Pi에 **Zoneminder** 설치


192.168.0.12/zm 에서 CCTV 화면 확인 가능

ZM - Monitor-1 - Feed - Chrome

주의 요함 | 192.168.0.12/zm/?view=watch&mid=2

Monitor-1 Scale: Actual Settings Close

Monitor-1 - 2018-12-01 17:48:18 +0900

A live video feed from a CCTV camera. The image shows a wooden door with a handle, set against a light-colored wall. The feed is displayed within a web browser window.

Disable Alarms State: Idle - 15.29 fps Force Alarm

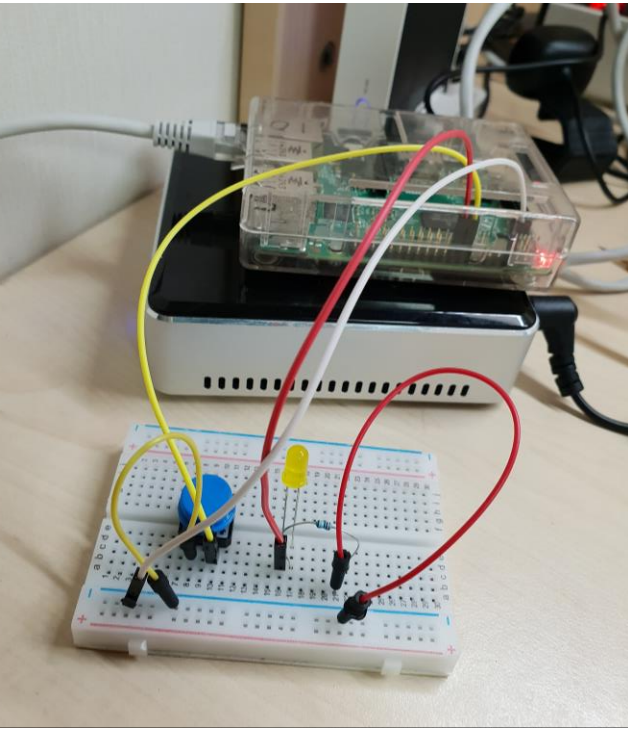
<< < || □ |> > >> -

Mode: Live Zoom: 1.0x

Id	Name	Time	Secs	Frames	Score
----	------	------	------	--------	-------

Progress details

택배함 통제



하드웨어 구성

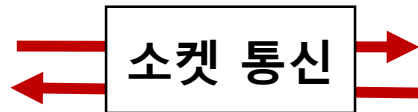
- 무인택배함 사용 요청은 스위치를 누르는 것
- 무인택배함의 승인은 led의 불이 들어오는 것
- 무인택배함을 닫는 것은 led의 불이 꺼지는 것

Progress details

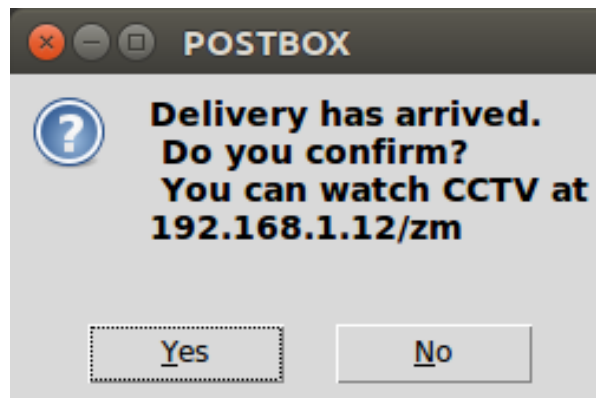
택배함 통제



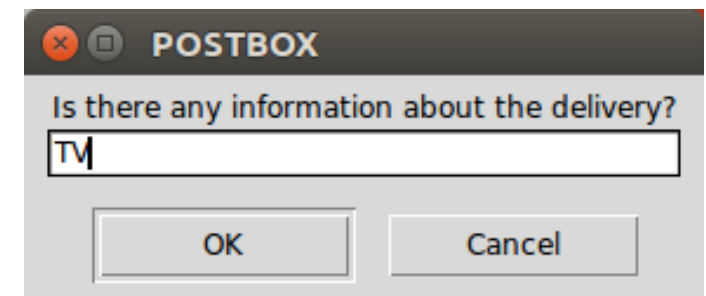
무인택배함
사용 요청



사용자는 CCTV 등을 확인하여
승인 또는 거절을 선택



<택배함 사용 요청>



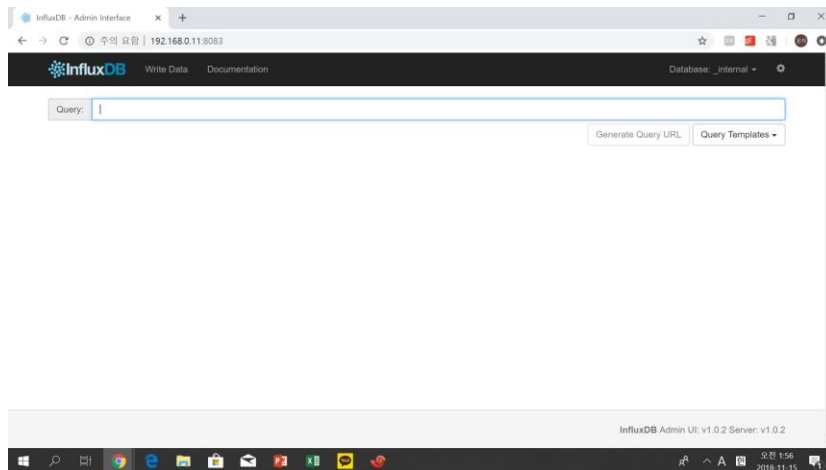
<승인하면 정보 기입>

Progress details

DB



도커를 설치한 후 컨테이너 안에 **influxDB**를 설치



192.168.0.11:8083 에서 **influxDB** UI 확인 가능

Progress details

DB



time(시간), 사용 요청(request), 승인(open), 거절(refused), 닫힘(closed), 택배 정보(information)

InfluxDB Write Data Documentation Database: POSTBOX ⚙		
Query: select * from Delivery		
Generate Query URL		Query Templates ▾
Delivery		
time	Status	information
2018-12-01T09:43:30Z	"request"	
2018-12-01T09:43:33Z	"Open"	"TV"
2018-12-01T09:44:53Z	"closed"	

Results

Results

결과



택배함 사용 요청이 들어오면,

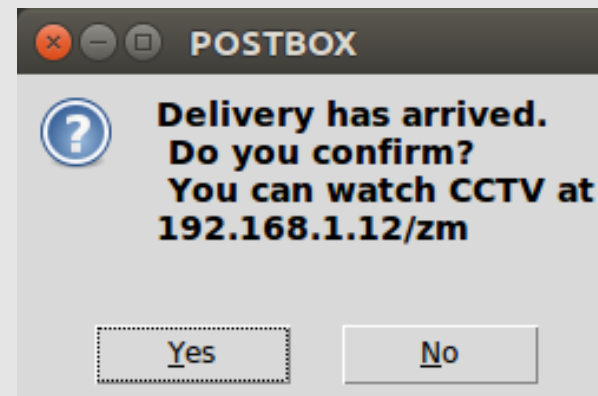
- ① DB에 요청(request) 기록을 저장
- ② 사용자(tower)에게 요청 전달

```
pi@raspberrypi:~ $ python pi.py  
complete DB write request
```



CCTV 등을 고려해 택배함 사용 승인 여부 결정

```
eun@eun:~/POSTBOX$ python db_client.py  
receive data from pi: Delivery has arrived
```

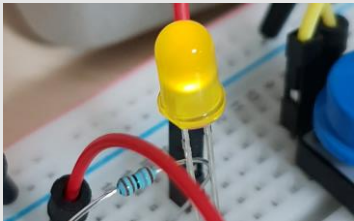


Results

결과



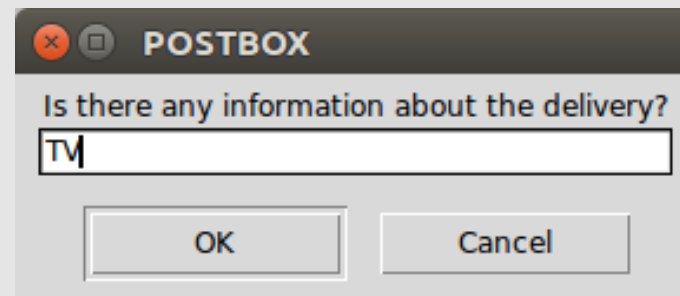
택배함 사용 가능



```
pi@raspberrypi:~ $ python pi.py  
complete DB write request  
True  
request agreed  
█
```



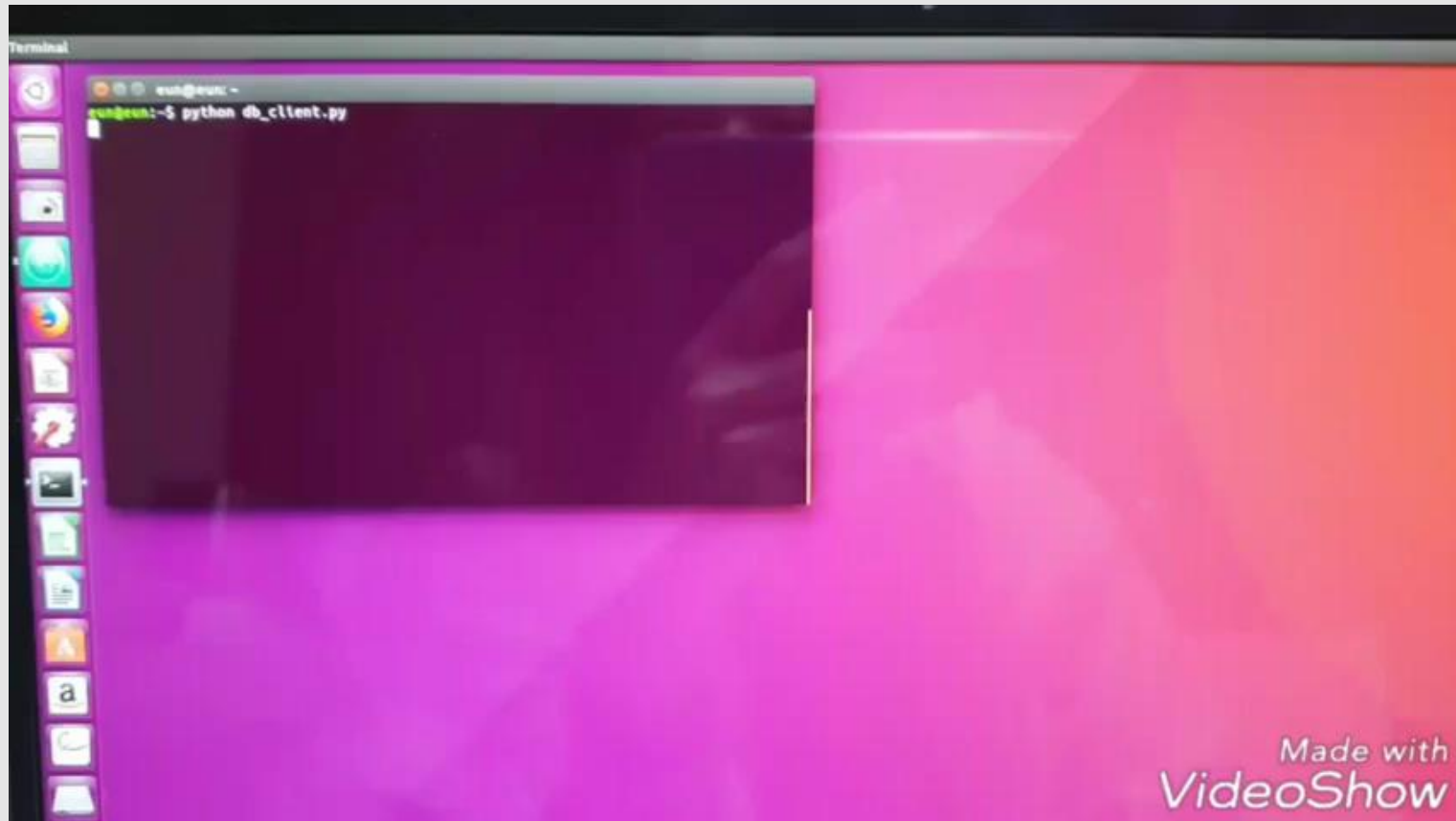
택배 정보 기입 가능



```
eun@eun:~/POSTBOX$ python db_client.py  
receive data from pi: Delivery has arrived  
Complete DB write status  
Complete DB write information  
█
```

Results

결과



Results

결과



택배함이 닫히면,
그 기록(closed)을 DB에 저장

```
pi@raspberrypi:~ $ python pi.py
complete DB write request
True
request agreed
Button pressed(closed)
```

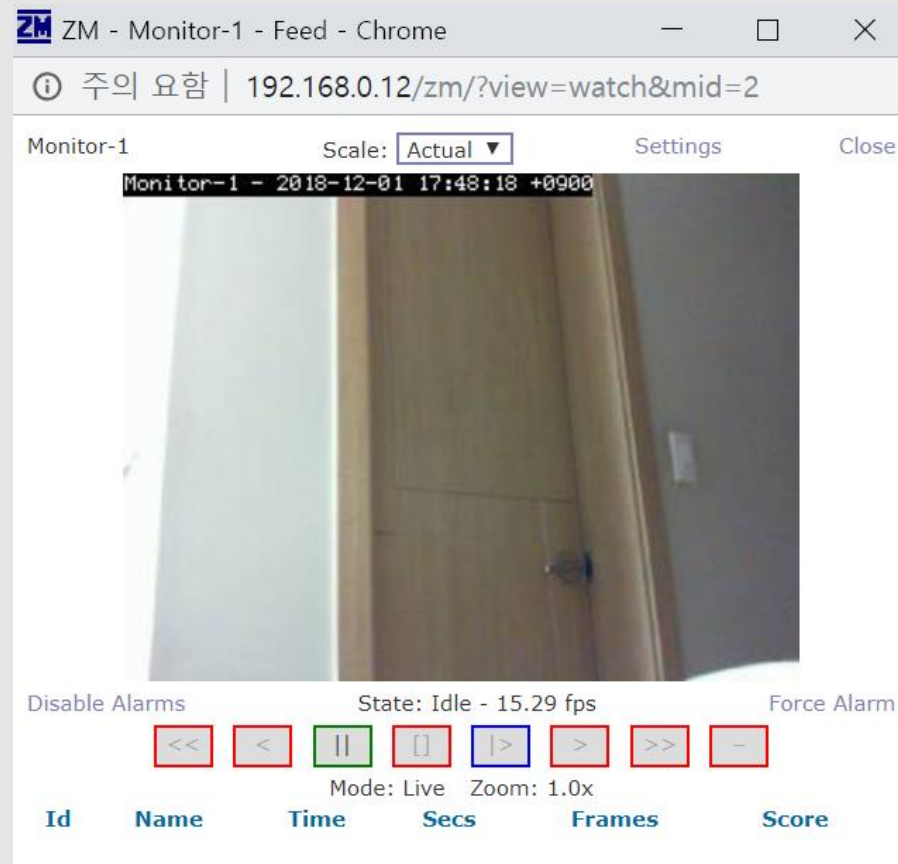


DB를 통해 택배함 사용 현황 관리

InfluxDB Write Data Documentation Database: POSTBOX		
Query: select * from Delivery		
Generate Query URL Query Templates		
Delivery		
time	Status	information
2018-12-03T18:02:42Z	"request"	
2018-12-03T18:02:55Z	"Open"	"TV"
2018-12-03T18:03:08Z	"closed"	

Results

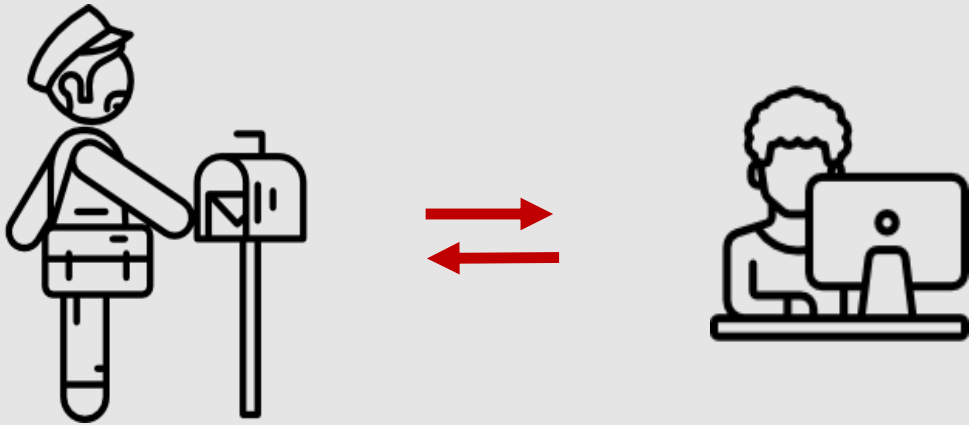
결과



택배함 앞을 촬영하고 있는
CCTV 확인 가능

Results

결과



무인택배함에 기사님이 도착했을 때,
방 안에서 CCTV 상황을 지켜본 후
무인택배함의 열고 닫음을 결정한다.
또한 무인택배함의 상황을 DB로 관리할 수 있다.

Complementary points

보완 사항



NUC이 사용자에게 도착하는 택배 정보를 알고 있다면?

Software lists

소프트웨어

tower.py

```
tower.py
1 import Tkinter
2 import tkMessageBox
3 import tkSimpleDialog
4 from datetime import datetime
5 from influxdb import InfluxDBClient
6 import socket
7
8 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9 server_socket.bind(('192.168.0.7',1113))
10
11
12 while True:
13     server_socket.listen(0)
14     client_socket, addr = server_socket.accept()
15     data = client_socket.recv(65535)
16     print("receive data from pi: " +data)
17     application_window = Tkinter.Tk()
18     result = tkMessageBox.askyesno("POSTBOX","Delivery has arrived.\n Do you confirm?\n You can watch CCTV at 192.168.1.12/zm \n", parent = application_window)
19     if result is True:
20         answer = 'Open'
21         send_data = 'True'
22     else:
23         answer = 'refused'
24         send_data = 'False'
25     current_time = datetime.utcnow().strftime('%Y-%m-%dT%H:%M:%SZ')
26     json_body = [ { "measurement" : "Delivery", "time" : current_time, "fields": { "Status" : answer } } ]
27     client = InfluxDBClient('192.168.0.11',8086)
28     client.switch_database('POSTBOX')
29     client.write_points(json_body)
30     print("Complete DB write status")
31
32     client_socket.send(send_data)
33
34     if result is True:
35         answer = tkSimpleDialog.askstring("POSTBOX", "Is there any information about the delivery?", parent=application_window)
36         json_body = [ { "measurement" : "Delivery", "time" : current_time, "fields": { "Status" : '', "information" : answer } } ]
37         client.write_points(json_body)
38         result2 = client.query('select information from Delivery;')
39         print("Complete DB write information")
40
41     application_window.destroy()
```

택배함이 부착된 pi로부터 요청이 들어오면,
승인 여부와 택배의 정보를 기입하고,
DB에 저장하고, pi로 승인 여부를 전달하는 파일

Software lists

소프트웨어

pi.py

Tower에 택배함 사용 요청을 전송하고, 승인 여부를 받고 이에 따라 택배함을 관리하는 파일

```
tower.py
1 from datetime import datetime
2 from influxdb import InfluxDBClient
3 import socket
4 import RPi.GPIO as GPIO
5 import time
6
7
8 def GPIO_set():
9     GPIO.setwarnings(False)
10    GPIO.setmode(GPIO.BCM)
11    GPIO.setup(21, GPIO.OUT)
12    GPIO.setup(23, GPIO.IN, pull_up_down = GPIO.PUD_UP)
13    GPIO.output(21, False)
14
15 def wait_button():
16     if GPIO.input(23) == 0:
17         print("Button pressed")
18         status_request()
19
20 def status_request():
21     current_time = datetime.utcnow().strftime('%Y-%m-%dT%H:%M:%S2')
22     json_body = [ { "measurement": "Delivery", "time": current_time, "fields": { "Status": "request" } } ]
23     client = InfluxDBClient('192.168.0.11', 8086)
24     client.switch_database('POSTBOX')
25     client.write_points(json_body)
26     print("complete DB write request")
27     TCP()
28
29
30 def status_closed():
31     current_time = datetime.utcnow().strftime('%Y-%m-%dT%H:%M:%S2')
32     json_body = [ { "measurement": "Delivery", "time": current_time, "fields": { "Status": "closed" } } ]
33     client = InfluxDBClient('192.168.0.11', 8086)
34     client.switch_database('POSTBOX')
35     client.write_points(json_body)
36     result2 = client.query('select Status from Delivery;')
37     print("Result: {}".format(result2))
```

```
pi.py
19 def TCP():
20     sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
21     sock.connect(('192.168.0.7', 1113))
22     output = 'Delivery has arrived'
23     sock.sendall(output.encode('utf-8'))
24     data = sock.recv(65566)
25     print(data)
26     if data:
27         print('request agreed')
28         status_open()
29         #switch on (5 sec)
30     else:
31         print('request refused')
32         #switch off
33
34 def status_open():
35     GPIO.output(21, True)
36     while True:
37         if GPIO.input(23) == 0:
38             print('Button pressed(closed)')
39             GPIO.output(21, False)
40             status_closed()
41             break
42         time.sleep(1)
43
44 if __name__ == '__main__':
45     GPIO_set()
46     status_request()
47     GPIO.cleanup()
```

Thank you :)