NDATOM

RNN과 군집로봇을 활용한 전기차 배터리 AI모니터링 시스템

Contents

Team

홍다윗
(팀장)

강민구
(로버구동파트)

김현준
(로버구동파트)

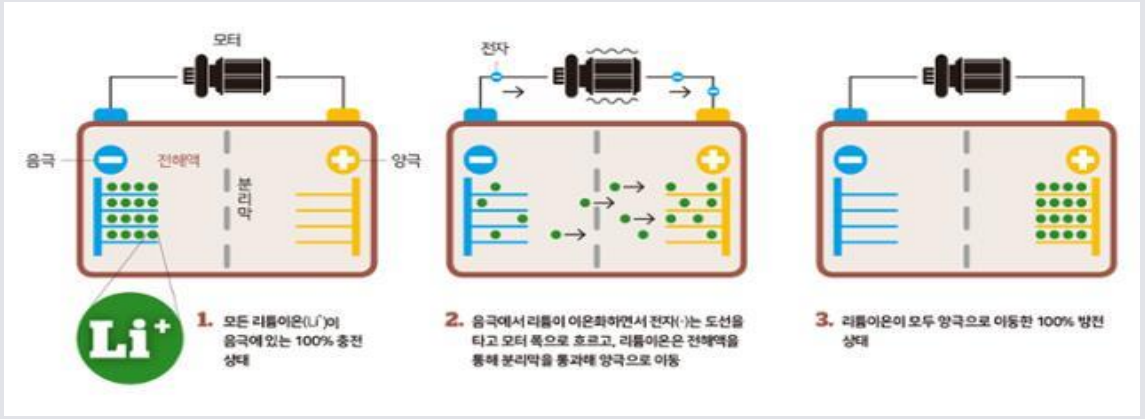
구명교
(센서 & 서버구축)

정상원(센서&서버구축)

주제 선정이유

전기차 화재의 위험성

리튬 이온 배터리를 쓰는 이유



리튬이온 배터리의 장점

1. 크기와 두께를 작고 가늘게 제작 가능
2. 고밀도로 에너지 저장 가능
3. 고전압까지 전압운용 가능

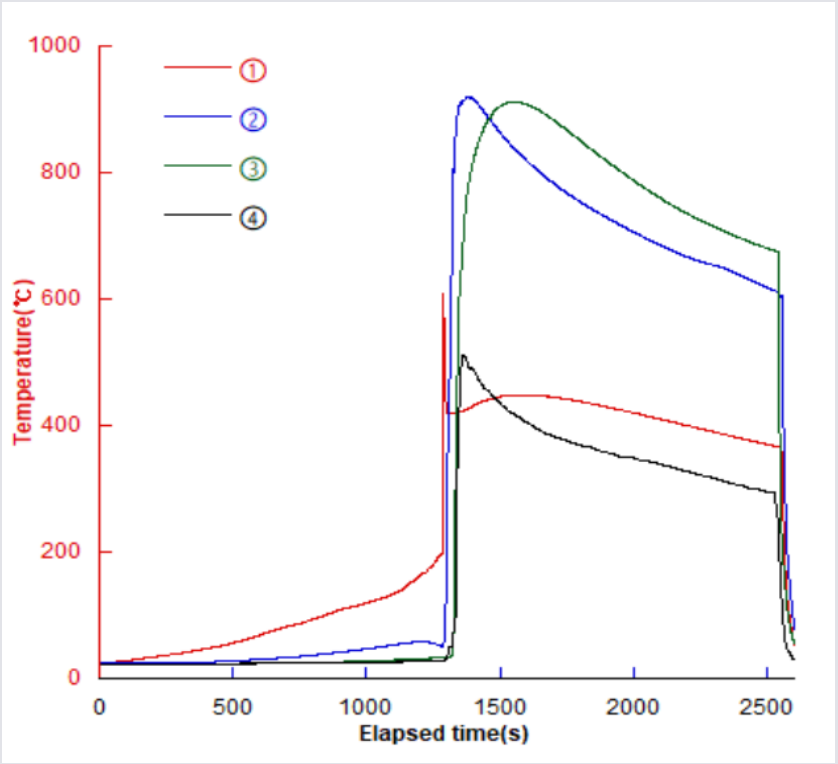
리튬이온 배터리의 단점

1. 열폭주 위험이 있음
2. 과방전시 용량감소가 매우 큼
3. 과충전시 매우 불안정 해짐

열폭주 온도가 비정상적으로 상승하며 화학 반응이 순차적으로 일어나는 연쇄 반응

Table 4. Thermal Runaway Time and Temperature

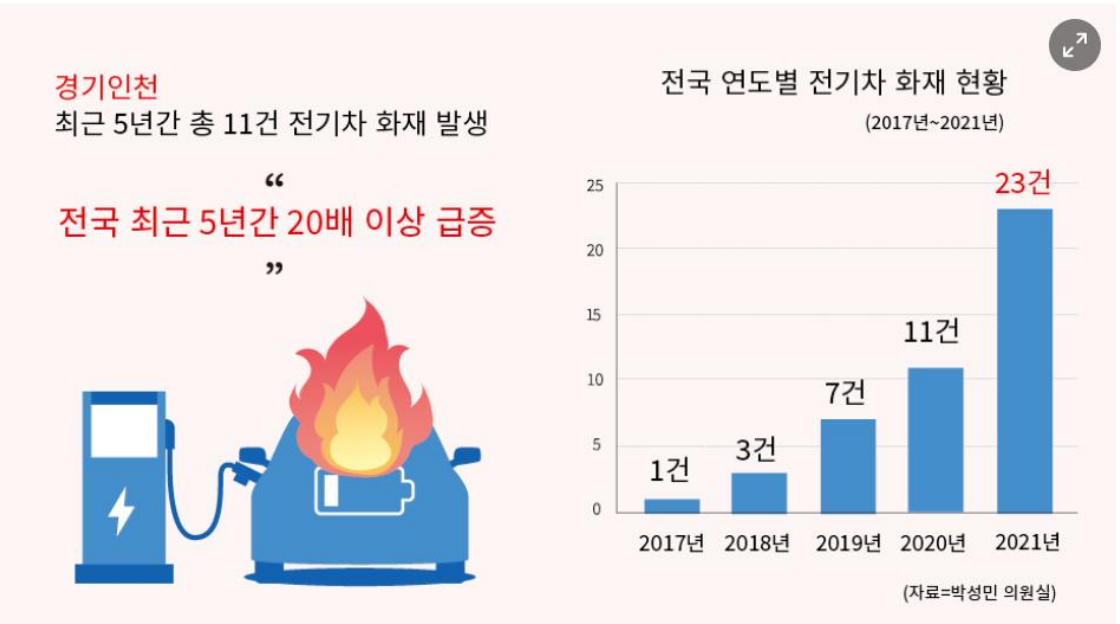
Experiment	Time to TR	Surface temperature	Temperature rise rate
1 st	21 min 23 s	212 ℃	8.0 ℃/min
2 nd	33 min 42 s	258 ℃	4.5 ℃/min
3 rd	22 min 13 s	246 ℃	7.6 ℃/min
4 th	22 min 48 s	215 ℃	7.4 ℃/min
5 th	26 min 13 s	259 ℃	7.9 ℃/min
6 th	29 min 18 s	258 ℃	4.9 ℃/min
7 th	26 min 17 s	280 ℃	5.9 ℃/min



분당 4~10도 범위로 온도를 상승 시켰을 때
최초 열폭주가 발생한 경과 시간은 **평균 26분**

급격한 온도 상승으로 약 **26분** 안에 대처해야한다.

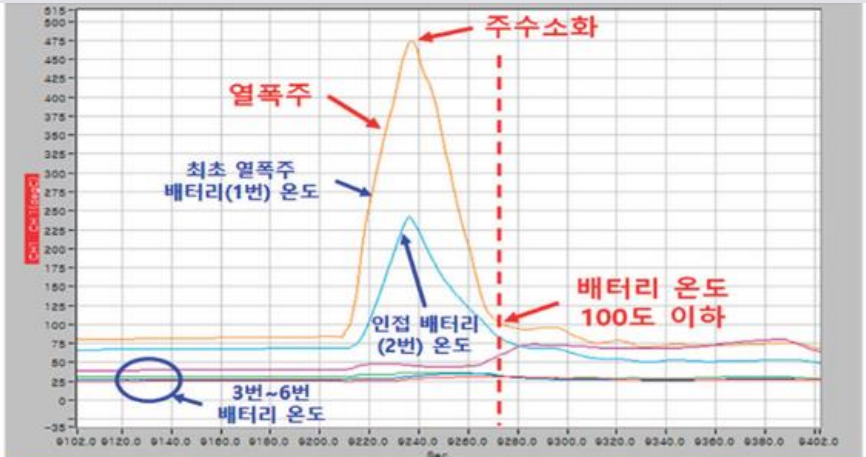
전기차의 고전압 리튬이온 배터리는 산소가 없어도 셀의 열폭주에 의해 불이 확산된다



현재 전기자동차 화재는 잘 꺼진다고 할 수 있는 대응 방법이 없다.
냉각소화나 질식소화덮개, 포소화약제를 활용한 질식 소화로 대응하고 있지만,
소화하기 어렵고 소화에 오랜 시간이 걸린다.

가장 효과적인 방법은
사전에 화재위험을 감지하는 방법이다.

리튬이온 배터리 열 폭주 현상과 소화 실험



배터리 최고 온도	475.2℃	배터리 온도 100℃ 이하로 냉각 소요시간	61초(주수)

프로젝트 주제

전기차화재 방지시스템 구축

작동 메커니즘

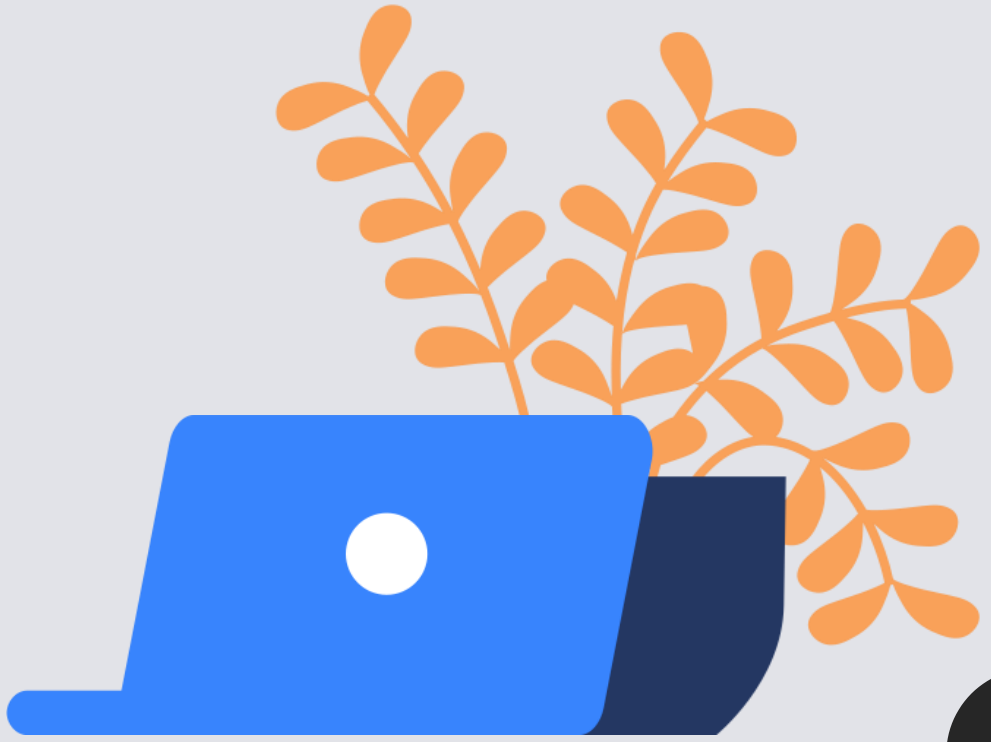
1.드론이 전기차 탐색

2.서버에 위치 전송

3.로버가 서버에서 좌표 값 받고

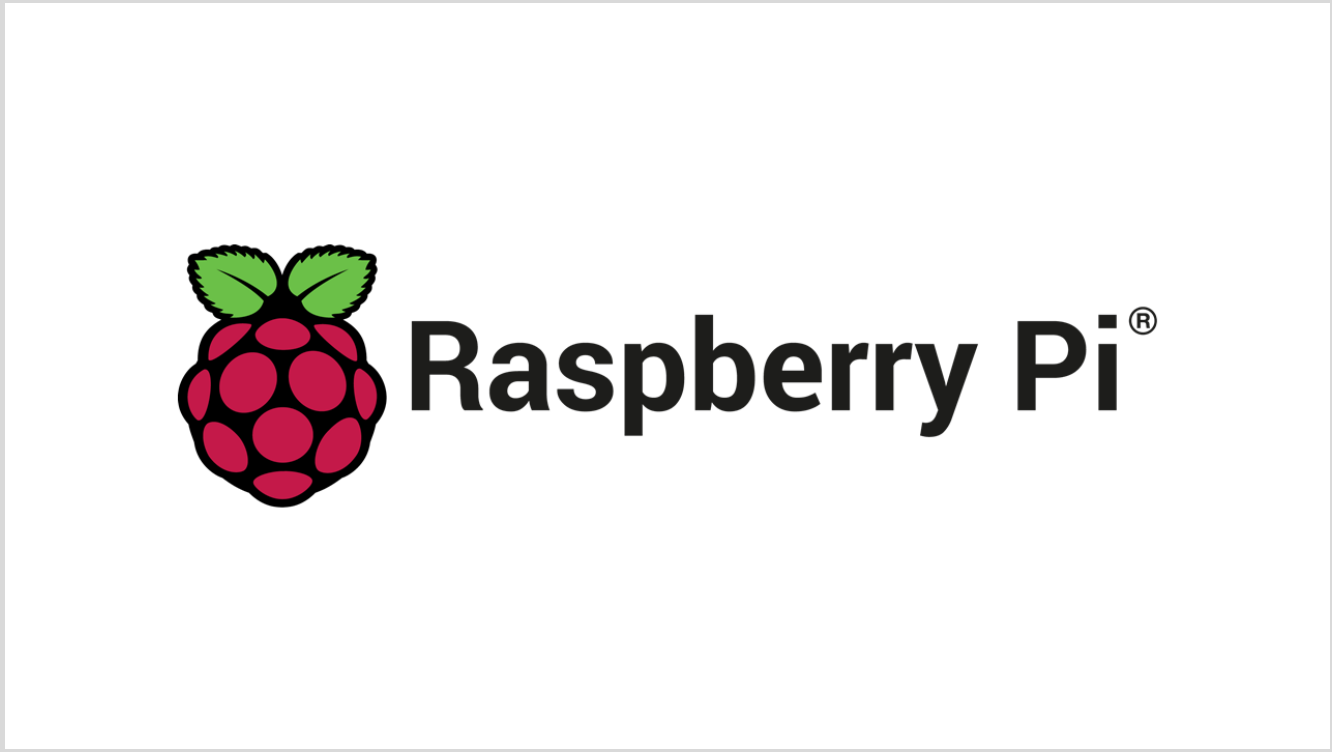
4.센서 측정 후 서버에 전송

5.서버에서 정보 처리



로버를 직접 개발한 이유

시중에 이미 바로 프로그래밍이 가능한 로버가 있지만, 해당 로버의 경우 판매사의 기기 사이의 통신 만 가능 등의 제한이 존재한다. 따라서 자율성을 위해 로버를 분해하고 소형 컴퓨터, 컨트롤러 등을 직접 연결하여 전기차 화재 탐지 전용 로버 개발하였다.



라즈베리 파이 역할

1. 파이썬으로 픽스호크 제어

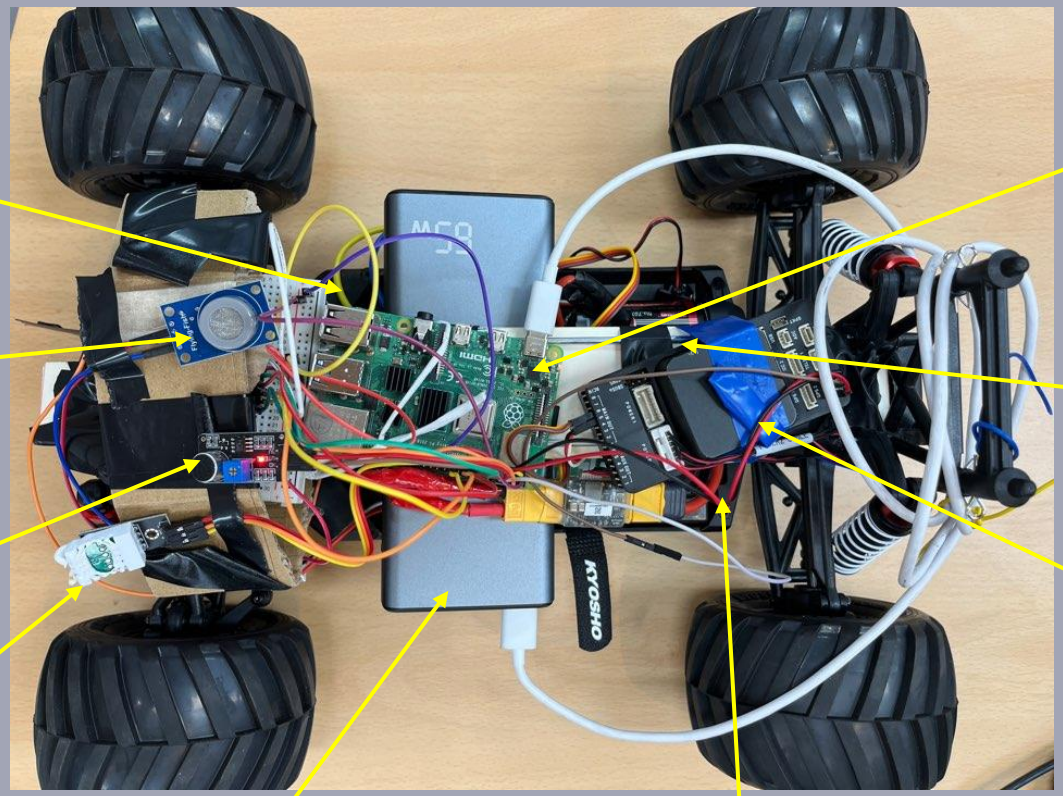
2. 측정 센서 제어

3. TCP/IP 통신 (로버와 서버 통신)

픽스호크 역할

1. 모터 제어
2. 로버 펌웨이 제공
3. 라즈베리 파이와 모터 연결

전기차 화재 탐지 전용 로버 개발



Servo 모터

이산화탄소
농도센서

소리 센서

온도센서

라즈베리파이 배터리

모터 및 픽스호크 배터리

라즈베리파이
(소형 컴퓨터)

ESC모터

픽스호크
(모터제어)

DRONE

주차장
비행

→

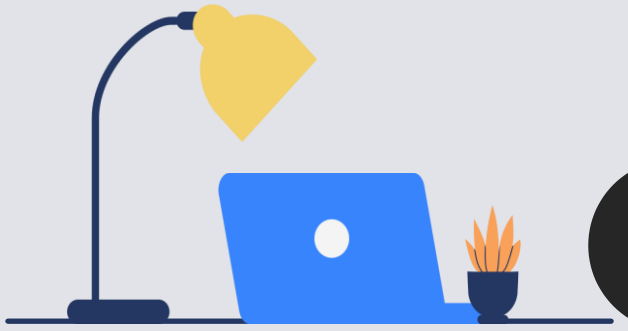
카메라와 Open cv를 통해
전기차 위치 확인

→

전기차 위치를
배열로 저장

→

해당 배열 txt 파일로
내보내어 서버에 저장



서버 구축

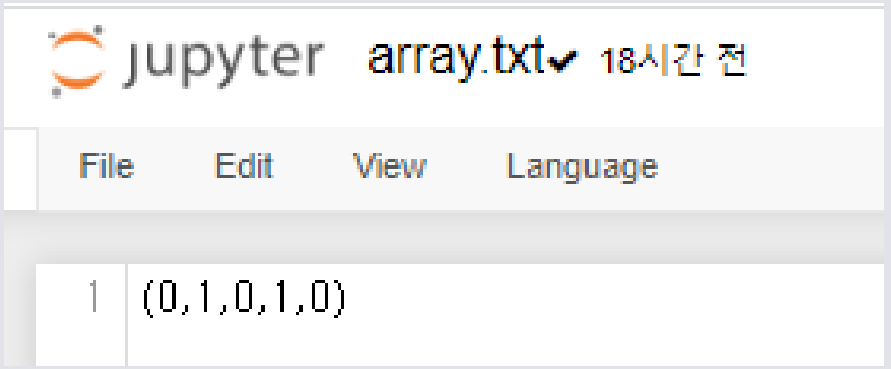
```python
import socket
import random
import numpy as np

# Change the port number
PORT = 12370

# Create a socket object
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to the port
s.bind(('', PORT))

# Listen for incoming connections
s.listen(1)

print('Server is running and waiting for connections...')

# Set the thresholds for temperature, humidity, and CO level
TEMP_THRESHOLD = 27
HUMIDITY_THRESHOLD = 39
CO_THRESHOLD = 36

# Read the array from the txt file
with open('array.txt', 'r') as f:
    array_str = f.read().strip('()\n')  # strip parentheses and newline characters
    array = [int(x) for x in array_str.split(',')]

while True:
    # Wait for a connection
    conn, addr = s.accept()

    print('Connected by', addr)

    # Send the array to the client
    conn.send(str(array).encode())

    # Receive the data in small chunks
    data = conn.recv(1024)

    if data:
        print('Received data: ', data.decode())

        # Parse the received data
        data = data.decode().split(',')
        temp = float(data[0].split(': ')[1][:-2])
        humidity = float(data[1].split(': ')[1][:-1])
        co = int(data[2].split(': ')[1])
        sound = int(data[3].split(': ')[1])

        # Check if the thresholds are exceeded or sound is detected
        if temp > TEMP_THRESHOLD and humidity > HUMIDITY_THRESHOLD and co > CO_THRESHOLD and sound == 1:
            print("There is a fire.")
```

서버 jupyter notebook 에서 실행

```
Server is running and waiting for connections...
Connected by ('192.168.0.5', 50176)
Connected by ('192.168.0.5', 50192)
Connected by ('192.168.0.5', 44420)
Connected by ('192.168.0.5', 44436)
Connected by ('192.168.0.5', 44448)
Connected by ('192.168.0.5', 44460)
Received data:  Temperature: 22.100000381469727° C, Humidity: 51.400001525878906%, CO level: 0, Sound: 0
Connected by ('192.168.0.5', 44462)
Connected by ('192.168.0.5', 58942)
Connected by ('192.168.0.5', 58954)
Received data:  Temperature: 22.200000762939453° C, Humidity: 51.5999984741210 94%, CO level: 0, Sound: 0
Connected by ('192.168.0.5', 58962)
Connected by ('192.168.0.5', 47530)
```

온도 일산화탄소 농도 소리

로버에서 보낸 측정 센서 값 서버에서 확인



드론에서 보낸 전기차 위치 배열

알고리즘

<열폭주 과정>

1. 배터리 내부의 SEI, 전해액 등의 분해
2. 일산화 탄소, 이산화 탄소 등 가스 발생
3. 53.7도에서 PCM이 액화되기 시작
4. 70도 이상에서 SEI층 파괴

↓

온도 위험 기준 설정

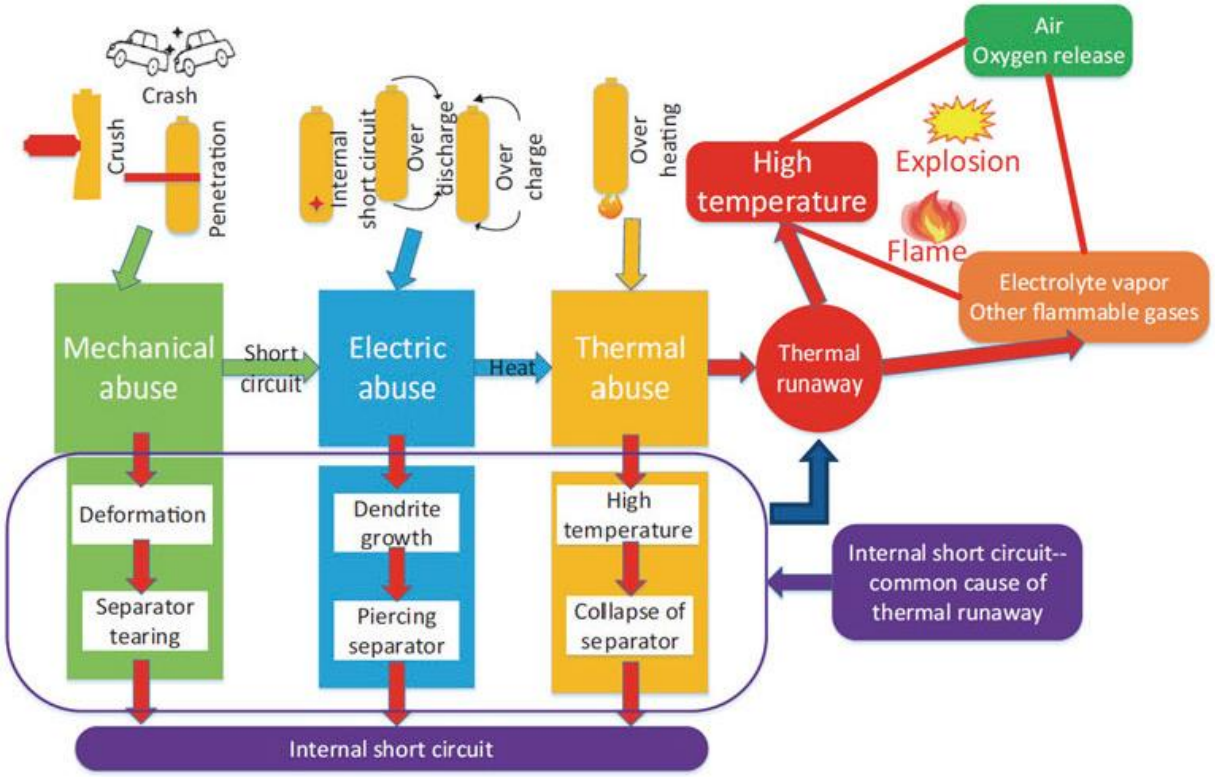
0~50도 : 1단계 (**정상**)

50~70도 : 2단계 (**경고**)

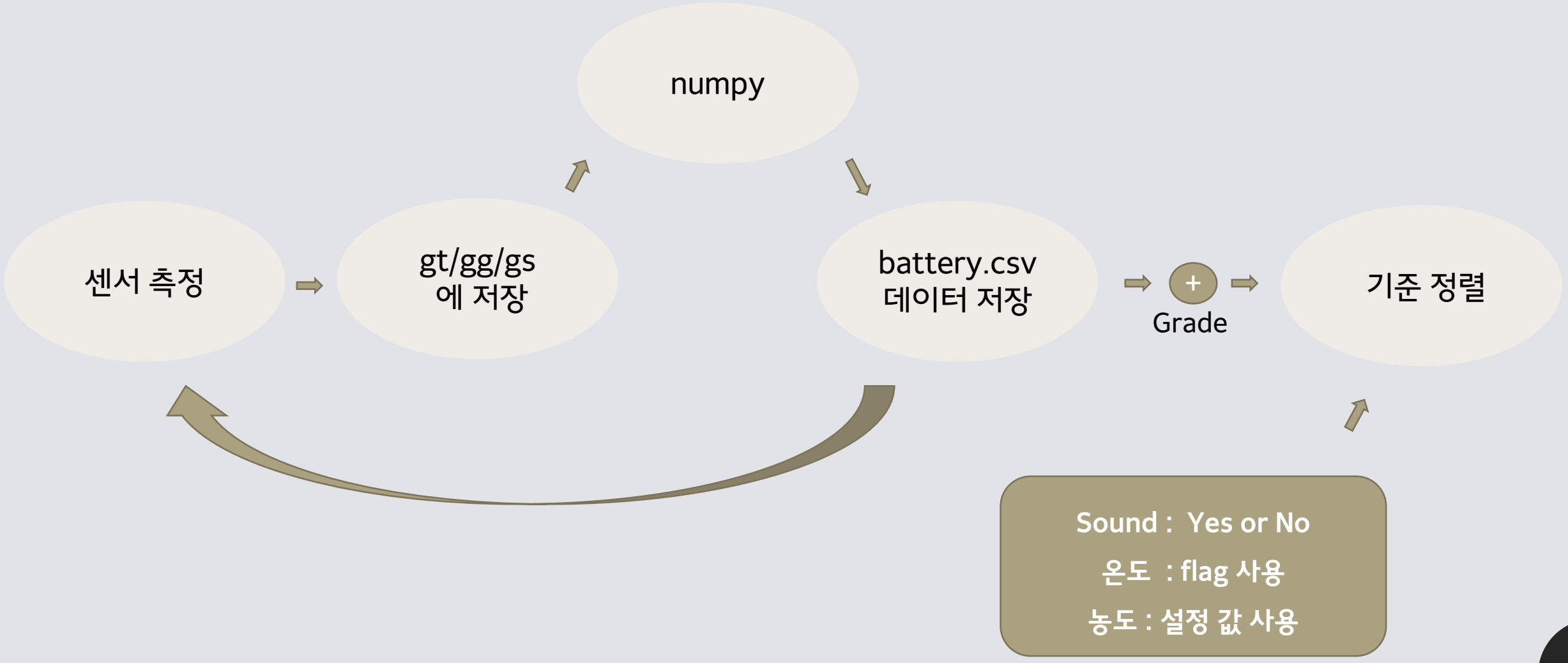
70도 이상 : 3단계 (**위험**)

배터리 화재에 대한 개략도

[그림 2] 배터리 화재 개략도 (Wang, 2019)



알고리즘 개략도


numpy
센서 측정
gt/gg/gs
에 저장
battery.csv
데이터 저장
+
Grade
기준 정렬
Sound : Yes or No
온도 : flag 사용
농도 : 설정 값 사용

1.각 센서 값 저장

```
# Test2
temp, co, sound, name = 40, 1, 0, "B"
if name not in flag:
    flag[name] = [0]  # flag를 리스트로 초기화
result = GT(temp, flag[name])
gt = result[0]
flag[name] = result[1]

gg = GG(co)
gs = GS(sound)
if name not in Car:
    Car.append(name)
subset_df = subset_df.append(pd.Series([temp, co, sound, name, gt+gg+gs, gt, gg, gs], index=subset_df.columns), ignore_index=True)

# Test3
temp, co, sound, name = 60, 1, 0, "C"
if name not in flag:
    flag[name] = [0]  # flag를 리스트로 초기화
result = GT(temp, flag[name])
gt = result[0]
flag[name] = result[1]

gg = GG(co)
gs = GS(sound)
if name not in Car:
    Car.append(name)
subset_df = subset_df.append(pd.Series([temp, co, sound, name, gt+gg+gs, gt, gg, gs], index=subset_df.columns), ignore_index=True)

# Test4
temp, co, sound, name = 35, 8, 0, "D"
if name not in flag:
    flag[name] = [0]  # flag를 리스트로 초기화

result = GT(temp, flag[name])
gt = result[0]
flag[name] = result[1]
```

2.데이터 축적 및 관리

```
온도 대피 KO
가스 대피 KO
가스 위험 D
```

	Temperature	Gas	Sound	Name	Grade	GT	GG	GS
0	0	0	0	Kim	3	1	1	1
1	70	11	0	KO	7	3	3	1
2	40	1	0	B	3	0	2	1
3	60	1	0	C	5	2	2	1
4	35	8	0	D	3	0	2	1

3. 분류 및 재정렬

In [61]:
```
#한 바퀴 돌고 재배열파트
# Step 1: Sort the dataframe by 'Grage' in descending order
sorted_df = subset_df.sort_values(by='Grade', ascending=False)

# Step 2: Filter rows that contain 'A', 'B', 'C', 'D' in the 'Name' column
filtered_df = sorted_df[sorted_df['Name'].isin(Car)]

# Step 3: Convert 'Name' column into an array
name_array = filtered_df['Name'].values
print("Priority")
print(name_array)
print("Input Car")
print(Car)

# Save 'name_array' to 'Car.txt'
# Save 'name_array' to 'Car.txt'
with open('Car.txt', 'w') as f:
    f.write(' '.join(name_array))
```

```
Priority
['KO' 'C' 'Kim' 'B' 'D']
Input Car
['Kim', 'KO', 'B', 'C', 'D']
```

4. Sound/온도/농도 센서값 처리

```
def GT(num, flag):
    if flag[0] > 0:
        if flag[0] == 1 and num < 45:
            return (1, [0])  # grade, flag
        elif flag[0] == 2 and num < 65:
            if(num>50):
                return(2,[1])
            else:
                return(1,[0])

    elif num >= 70:
        print("온도 대피 " + name)
        return (3, [2])  # grade, flag
    elif num>=50:
        return(2,[1])
    return (0, [1])  # grade, flag
```

```
def GG(num):
    if num>10:
        print("가스 대피",name)
        return 3
    elif num>5:
        print("가스 위험",name)
        return 2
    else:
        return 2

def GS(num):
    if(num>0):
        return 2
    else:
        return 1

# Upload files from your local directory
```

앞으로 발전 가능성

정확성

로버가 직접 전기차 아래로 들어가 배터리 관련 정보를 측정하므로 정확도 **상승**

비용의 효율성

각 주차 장소마다 센서가 필요한 기존의 스마트 주차장과 달리 움직이는 로버에만 센서가 있으므로 비용 **감소**

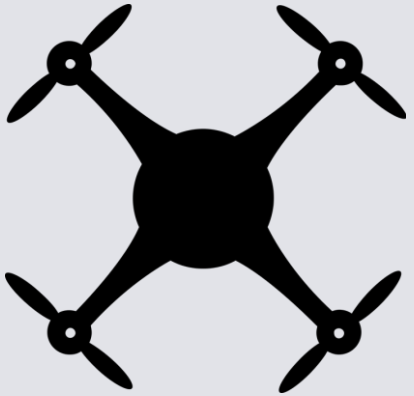
편의성

어디든 설치가 가능하므로 행사장 임시 주차장, 야외 주차장 등 에 설치하여 주차장 내 전기차 화재관리가 가능

유사 사례와 차별화

센서 사용 감소로
경제성 확보

모바일 기술 활용
넓은 영역 적용가능

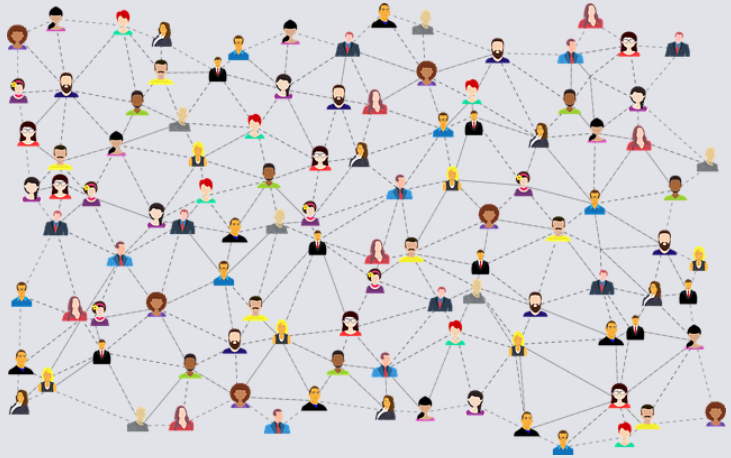


드론과 로버를 통해
정보 수집 및 분석

예상수요

지속가능성에 대한 관심이 높아짐에 따라 전기차에 대한 수요 증가

목표시장



전기차가 존재하는 모든 시설

쇼핑몰, 공장, 공항, 대도시 등

목표계획

아주대학교 축제 기간에 학교와

연계를 목표로 하며 이를 시작으로

수원시나 도청행사 등에 지원

NDATOM

우리의 목표는

전기차 화재로부터 안전한 환경을

지속적으로 제공하는 일입니다.