Busan science high school
2023 Ocean ICT Festival
2023 BOIF
C 06

QR코드 영역
QR 삽입 후
테두리 삭제

Youtube 영상 QR

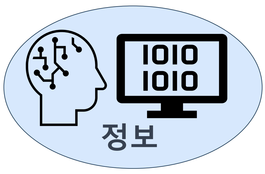
AI와 아두이노를 이용한 재생 에너지 발전소의 에너지 저장 및 분배 최적화

팀명: Energy Safer
2103 김정현, 2106 조민경

연구 동기

지속가능한 에너지 원천인 해양 에너지의 가치를 극대화하기 위한 방법을 찾아보려 했다. 특히 해양 에너지의 경우, 환경 조건의 변동성이 크기 때문에 AI를 이용한 동적인 에너지 관리 방법을 도입하는 것이 효과적이라고 생각했다. 이 활동을 통해, AI와 기계학습을 활용하여 해양 에너지의 효율성을 향상시키고, 그 상업적 가치를 높일 수 있을 것이다.

융합분야



현재의 환경 데이터를 바탕으로, AI는 앞으로 변화할 환경 조건을 예측하고, 그에 따른 미래의 에너지 생성량을 예상한다. 이런 예측 결과를 기반으로 현재의 **에너지 분배 그리드**를 조정한다.

코드

```python
import requests
import numpy as np
import random
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# LSTM 모델을 위한 설정
n_steps = 5
n_features = 1

# LSTM 모델 생성 함수
def create_lstm_model():
    model = Sequential()
    model.add(LSTM(50, activation='relu', input_shape=(n_steps, n_features)))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    return model

house_model = create_lstm_model()
plant_model = create_lstm_model()

# OpenWeatherMap으로부터 날씨 데이터 가져오기
def get_weather_data(city):
    API_KEY = 'YOUR_OPENWEATHERMAP_API_KEY'
    base_url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={API_KEY}"

    response = requests.get(base_url)
    if response.status_code == 200:
        data = response.json()
        temp = data["main"]["temp"]
        humidity = data["main"]["humidity"]
        return temp, humidity
    else:
        print(f"Error {response.status_code}: Unable to fetch weather data.")
        return None, None

def predict_power_consumption(model, previous_data):
    x_input = np.array(previous_data[-n_steps:])
    x_input = x_input.reshape((1, n_steps, n_features))
    yhat = model.predict(x_input, verbose=0)
    return yhat[0][0]

def update_internal_systems_by_ai(previous_data):
    predicted_power_needs = {}
    for system, data in previous_data.items():
        predicted_power_needs[system] = predict_power_consumption(plant_model, data)
    return predicted_power_needs

def main():
    # 사용자 입력
    season = input("현재 계절을 입력하세요 (spring, summer, autumn, winter): ")  # 사용자 입력
    city = input("도시 이름을 입력하세요 (예: Seoul): ")

    temp, humidity = get_weather_data(city)
    if temp and humidity:
        print(f"Temperature: {temp}K, Humidity: {humidity}%")

    # 가정별 전력 사용 기록
    house_data = [[random.randint(10, 100) for _ in range(n_steps)] for _ in range(5)]
    predicted_house_power = [predict_power_consumption(house_model, data) for data in house_data]

    # 발전소 내부의 전력 사용 기록
    internal_data = {
        "control_room": [random.randint(10, 50) for _ in range(n_steps)],
        "turbine_generator": [random.randint(50, 100) for _ in range(n_steps)],
        "cooling_system": [random.randint(10, 60) if season == "summer" else random.randint(10, 40) for _ in range(n_steps)],
        "storage_system": [random.randint(10, 50) for _ in range(n_steps)]
    }
    predicted_internal_power = update_internal_systems_by_ai(internal_data)
```

{날씨 및 계절 고려}
날씨 데이터는 OpenWeatherMap API를 통해 특정 도시의 현재 온도와 습도 정보를 가져왔다. 계절 정보는 사용자로부터 입력을 받아 처리한다.
*계절별로 전력 소비 패턴이 달라질 수 있기 때문에 중요한 정보이다.

{집의 상태 (ON, OFF)}
각 집의 전력 상태는 사용자로부터 "ON" 또는 "OFF"로 입력을 받는다.
"ON"상태의 집은 일정량의 전력을 사용하며, "OFF"상태의 집은 전력을 사용하지 않는다.

{발전소 내부 전력 소비}
발전소도 일정량의 전력을 소비한다.
발전소 내부의 전력 소비량은 LSTM 모델을 통해 예측하고 관리한다.

{LSTM 딥러닝 모델 사용}
이 모델은 과거 전력 사용 패턴을 학습하여 미래의 전력 소비량을 예측한다. 코드에는 집과 발전소 내부 시스템에 대한 각각의 LSTM 모델이 구현되어 있다.

{실시간 업데이트}
사용자는 프로그램을 실행하는 동안 여러 번 전력 상태를 변경할 수 있다.
(예: 집의 전력 상태 변경)
매번의 변경마다 AI는 이를 학습하고, 전력 분배를 실시간으로 업데이트하며 결과를 출력한다.
이러한 요소와 처리 방법을 통해, 코드는 실시간으로 전력 분배를 최적화하고 결과를 사용자에게 제공한다.

결론 및 고찰

이 프로젝트를 통해 발전소 내부와 주변 집들의 전력 사용량을 **AI**를 통해 최적화한 스마트 그리드 시스템을 구현할 수 있었다. **LSTM**을 활용하여 시계열 데이터인 전력 사용량을 예측하였고, 사용자의 입력에 따라 실시간으로 전력 분배를 조절할 수 있었다. 또한 **AI**의 자동화를 이용해 다양한 변수를 고려할 수 있었다. 프로젝트는 5개의 집을 대상으로 했지만, 실제 스마트 그리드 시스템에서는 수백, 수천의 건물과 장치를 관리해야 한다. 이를 위해서는 시스템의 확장성과 효율성을 이후 더 업그레이드해야 한다. 또한 날씨와 계절 외에도 전력 사용량에 영향을 미치는 다양한 외부 요인들을 고려하여 모델을 더욱 정교하게 만들어야 한다.