**3.2 IoT 빅데이터 수집 및 처리 설계.**

태양광 관련 데이터는 기관에서 공개를 하지 않고 설사 공개하더라도 돈이 필요해 웹에서가 아닌, 3.1절에서 말한 IoT 센싱 디바이스를 설계하여 디바이스에서 전송한 정보를 받아 처리하고, 저장하는 것이 필요하다. Node.js는 확장성 있는 네트워크 애플리케이션(특히 서버 사이드) 개발에 사용되는 소프트웨어 플랫폼이다. 작성 언어로 자바스크립트를 활용하며 Non-blocking I/O와 단일 스레드 이벤트 루프를 통한 높은 처리 성능을 가지고 있다. 또한 MongoDB는 빅데이터 처리에 가장 적합한 NoSQL 기반으로 쿼리가 매우 쉽게 변환되기 때문에 관계형 데이터베이스를 MongoDB로 쉽게 변환할 수 있는 강력한 문서 지향 쿼리 언어에 있다. 본 연구에서는 Node.js로 센싱 디바이스가 보내는 정보를 받아 MongoDB에 저장하는 서버 시스템을 구성하였다.

■ 서버 설계

**서버 구성**

Node.js는 확장성 있는 네트워크 애플리케이션(특히 서버 사이드) 개발에 사용되는 소프트웨어 플랫폼이다. 작성 언어로 자바스크립트를 활용하며 Non-blocking I/O와 단일 스레드 이벤트 루프를 통한 높은 처리 성능을 가지고 있다.

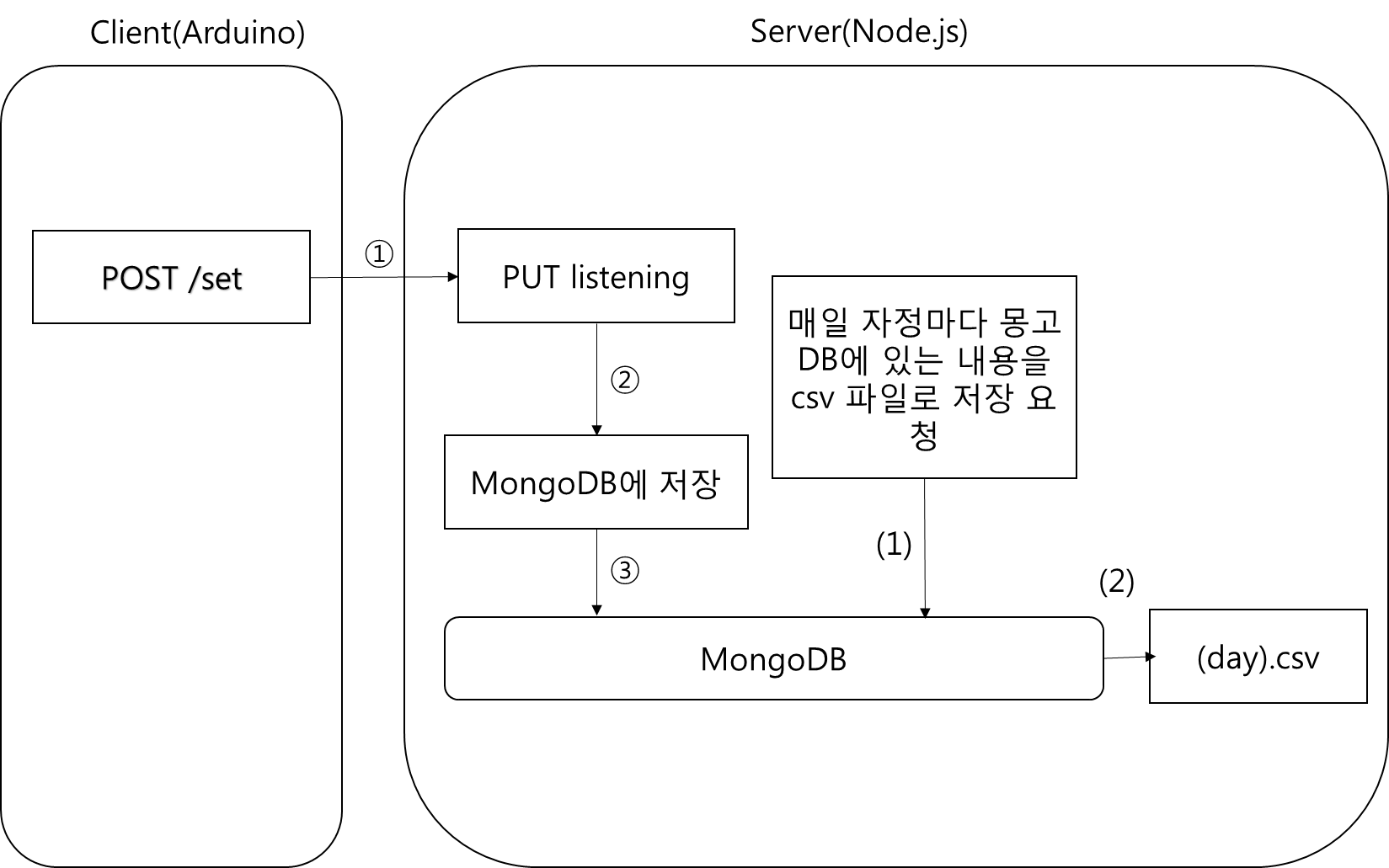
내장 HTTP 서버 라이브러리를 포함하고 있어 웹 서버에서 아파치 등의 별도의 소프트웨어 없이 동작하는 것이 가능하며 이를 통해 웹 서버의 동작에 있어 더 많은 통제를 가능케 한다.

또한 Node.js는 npm라는 자바스크립트 프로그래밍 언어를 위한 패키지 관리자를 기본 패키지 관리자로 들고 있다. npm으로 인해 버전 관리, 라이브러리 관리 등 서버를 구성할 때 필요한 기능들을 쉽게 확장할 수 있게 해준다.

서버 시스템의 기능은 아래와 같이

1. 디바이스로부터 날라오는 데이터를 받아 처리하는 서버
2. 데이터를 받아 저장하는 DB
3. 자정이 지나면 DB에 있는 내용을 파일로 변환하는 백업 시스템

로 구성된다. 이를 간단하게 그림으로 나타내면 아래 <그림 x> 가 된다.



<그림 1. 서버 구성도>

1)번 기능은 npm에서 제공하는 coap 모듈 다운로드 할 필요가 있으며 2)번 기능은 npm에서 제공하는 mongodb 모듈을 다운로드 하고 MongoDB 서버를 설치하여야한다. 그리고 3)번 기능은 mongo-to-csv , node-schedule 모듈을 설치해야 한다.

최종적으로 서버 구성에 필요한 프로그램 , 모듈은 다음과 같다.

⊙ Node.js

1. CoAP 패킷을 처리할 수 있게 해주는 coap 모듈
2. MongoDB에 삽입하고 검색할 수 있게 해주는 mongodb 모듈
3. MongoDB의 데이터를 csv 파일로 추출하는 mongo-to-csv 모듈
4. 특정 시간대마다 함수를 실행시키는 node-schedule 모듈

⊙ MongoDB Server

**서버 기능 설계**

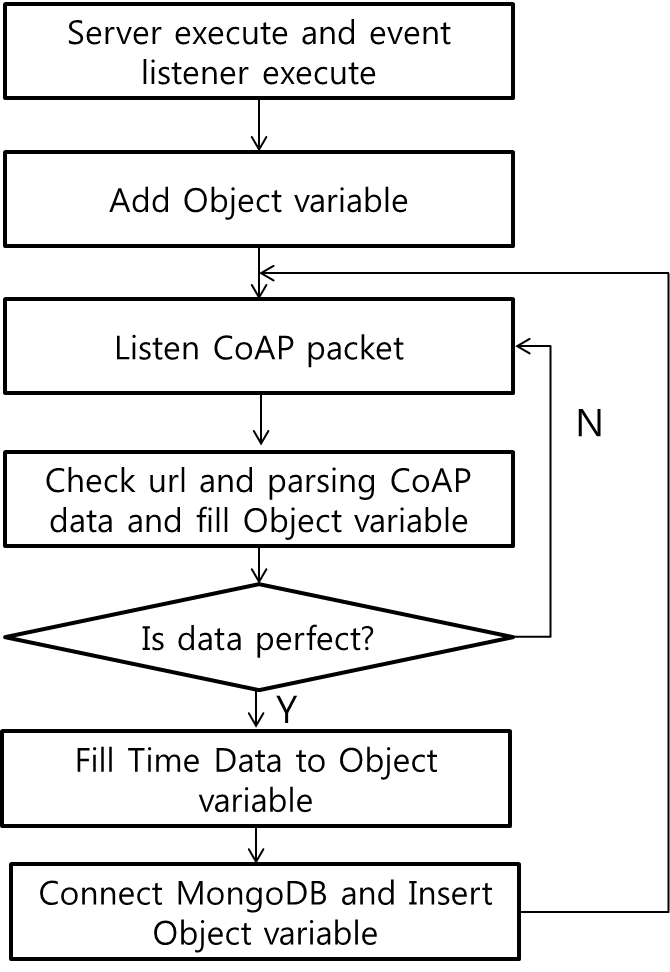
Node.js 서버는 크게 두 가지 기능으로 구분되는데 첫째는 CoAP 데이터를 받고 MongoDB에 저장하는 기능 , 둘째는 데이터 분석을 위해 특정 시간마다 MongoDB에 있는 내용을 csv 파일로 추출하는 것이다. 각각의 기능은 밑에 따로 서술하였다.

**1.CoAP 수신 기능 설계(<그림 1>의 ① , ② , ③**)

Node.js 서버에서는 주기적으로 데이터를 보내는 아두이노로부터 CoAP 데이터를 받아 DB 관리, 백업 파일 추출에 필요한 필드값들을 추가하고 MongoDB에 저장하는 역할을 수행해야 한다. 위 <그림 1>의 서버 구성도 중에서 ① , ② , ③의 동작을 수행하는 기능이다. 이를 수행하기 위해 Node.js 서버는 아래의 기능을 순차적으로 반복 수행한다.

1. CoAP 패킷 수신
2. 패킷이 손실 없이 들어왔는지 확인
3. 4절의 데이터 분석에 필요한 시간값과 백업 파일 추출에 필요한 필드값을 추가
4. MongoDB에 접속 및 삽입

이상에서 설명한 4가지 주요 기능을 수행하는 서버의 동작을 아래 <그림 x>에 순서도를 이용하여 나타내었다.



<그림 x>. CoAP 수신 기능 동작 흐름도>

**2. csv 파일로 추출 백업 기능 설계(<그림 1>의 (1) , (2))**

매일 자정이 되면 MongoDB에서 특정 날짜에 해당하는 데이터들을 csv 파일로 저장한다. 위 <그림 1>의 서버 구성도 중에서 (1) , (2)의 동작을 수행하는 기능이다. Node.js 서버는 아래의 기능을 순차적으로 반복 수행한다.

1. 자정이 되면 스케쥴러 함수 실행
2. 현재 날짜 얻기
3. MongoDB 에 있는 데이터 중에서 특정 날짜의 데이터를 csv 파일로 추출.

이상에서 설명한 3가지 주요 기능을 수행하는 서버의 동작을 아래 <그림 x>에 순서도를 이용하여 나타내었다.

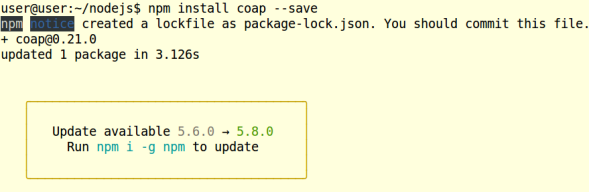


<그림 x>.추출 백업 기능 흐름도>

■ 서버 동작 상세 설명

ㅇ CoAP 패킷 수신

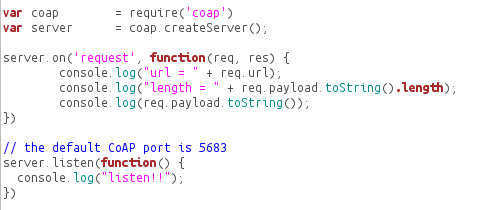
Node.js가 기본적으로 제공하는 라이브러리로는 CoAP 패킷을 처리할 수가 없다. 따라서 CoAP를 처리할 수 있게 하기 위해서는 위에서 상술한 npm(node package manager)에서 coap라는 패키지를 설치해야 한다. 설치하는 방법은 리눅스 명령어로 npm install coap –save 를 치면 된다.



<그림>

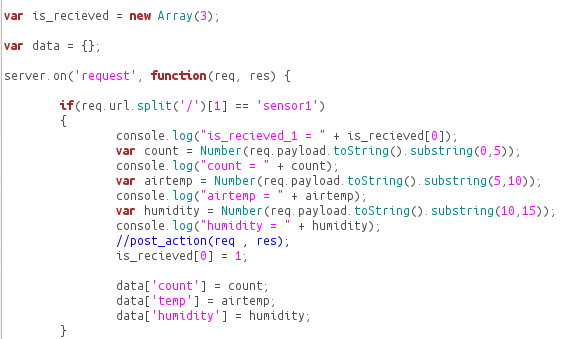
패키지를 설치하면 위 <그림>과 같이 설치가 됐다는 텍스트가 뜨며 node\_module 디렉토리가 생기고 그 안에 coap 관련 모듈이 들어가게 된다. Coap 모듈의 API 문서 및 샘플 소스는 <https://github.com/mcollina/node-coap> 웹페이지에서 볼 수 있다.

밑의 <그림 x>는 CoAP 데이터를 받을 수 있는 기본적인 서버를 실행한 코드이다. coap 변수로 ‘coap’ 모듈을 로드하고 server 변수로 서버를 실행함으로써 server 변수가 CoAP 데이터를 받을 수 있게 된다. server.listen 메소드는 서버가 실행될 때 수행되는 함수이며 server.on 메소드는 특정 이벤트가 발생했을 때 수행되는 이벤트 핸들러다. 함수의 첫번째 매개변수의 ‘request’ 매개변수를 집어넣어 request 데이터가 오면 수행되게 되어있다.

****

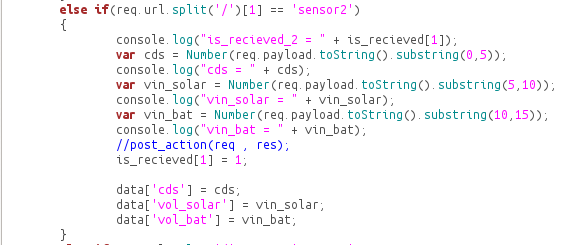
ㅇ 수신된 데이터 파싱 및 결합

아두이노의 문제로 CoAP의 POST 기능은 사용이 불가능해 PUT으로 대체하게 되었다. 단, 실험해 본 결과 PUT 기능은 데이터로 20byte 정도밖에 보낼 수 없어 아두이노에서 생성되는 데이터인 데이터를 전부 보낼 수 없다. count , 온도(temp) , 습도(humidity) , 조도(cds) , 태양판에서 생성된 전압(vin\_solar) , 배터리에 남아있는 전압(vin\_bat) , 태양판에서 생성된 전류(amps\_solar) , 배터리에 남아있는 전류(amps\_bat) 총 8개의 데이터는 각각 5byte씩 총 40byte를 차지한다. 이는 PUT 메세지가 한번에 전송할 수 있는 20byte를 초과하는 데이터양으로 이를 해결하기 위해 메세지를 3번을 나눠서 보냈다. 각 메세지를 받을때마다 is\_recieved 배열의 각 숫자에 해당하는 인덱스를 1로 바꾸고 위에서 선언한 data라는 객체로 삽입한다.



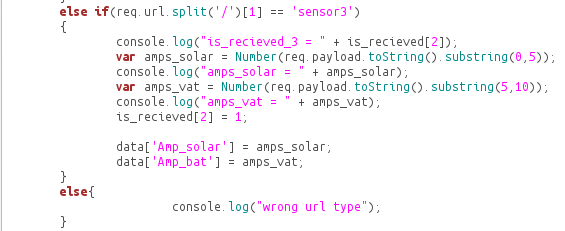
<그림 x>

위의 <그림 x>는 위에서 얘기한 8개의 필드값 중에 앞의 3개인 count , 온도 , 습도 데이터가 담긴 sensor1 값을 받아 데이터를 파싱하고 sensor1 값을 받았다는 표시로 is\_recieved[0] 변수를 1로 만든 다음, MongoDB에 담을 data 객체에 데이터를 넣는 코드이다.



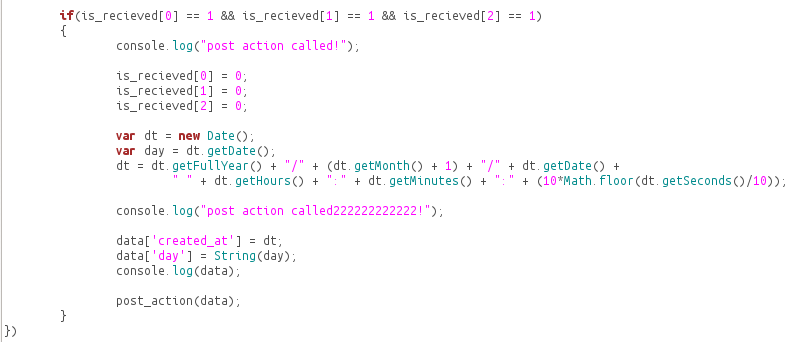
<그림 x>

위의 그림도 마찬가지로 아두이노에서 보낸 데이터가 들어오는 코드이다. sensor2 값으로 조도 , 태양판넬이 생성한 전압 , 배터리 전압의 정보를 파싱해서 얻은 다음, sensor2 데이터를 받았다고 is\_recieved[1] 변수를 1로 만든 다음 data 객체에 값을 넣는다.



<그림 x>

<그림 x>는 마지막 값들인 태양광 판넬에 흐르는 전류, 배터리 전류의 값을 받아온 것이다. 마찬가지로 sensor3 데이터를 받으면 is\_recieved[2] 변수를 1로 만들고 data 객체에 값을 집어넣는다.

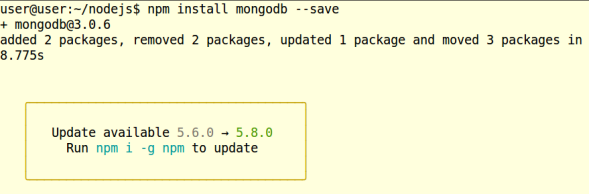


<그림 x>

<그림 x>는 CoAp 데이터가 들어오면 sensor1 , sensor2 , sensor3 값들이 모여 온전한 데이터를 만들 수 있는지를 검사하고 이 값들이 다 들어와 온전한 데이터를 만들 수 있으면 체크변수들을 다 0으로 초기화하고 데이터 분석에 필요한 시간값 , csv 파일로 추출하기 위한 날짜를 객체에 추가시킨 다음 post\_action 함수를 실행시킨다. 이 때 매개변수로 아두이노로부터 받은 데이터들을 담고 있는 data 변수를 넘긴다.

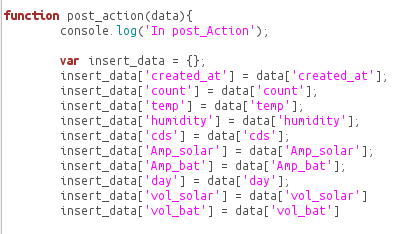
ㅇ MongoDB로 데이터 삽입

Node.js가 MongoDB에 접속하고 데이터를 삽입, 탐색하려면 Node.js에서 돌아가는 MongoDB 드라이버를 설치해야한다. 이는 npm install mongodb --save 명령으로 다운로드 할 수 있다.



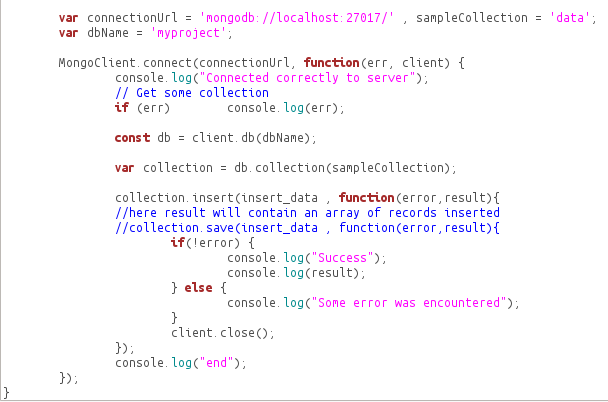
<그림>

패키지를 설치하면 위 <그림>과 같이 설치가 됐다는 텍스트가 뜨며 node\_module 디렉토리가 생기고 그 안에 mongodb 관련 모듈이 들어가게 된다. MongoDB 모듈의 설치, API 문서 및 샘플 소스는 <https://www.npmjs.com/package/mongodb> 에서 볼 수 있다. <http://mongodb.github.io/node-mongodb-native/3.0/> 은 조금 더 자세한 API , 특정 버전의 API를 볼 수 있다.



<그림 x>

위의 ‘수신된 데이터 파싱 및 결합’ 부분에 있는 post\_action이 실행되면 MongoDB로 데이터의 삽입이 이루어진다. 위의 <그림 x>는 함수가 수행되고 insert\_data라는 객체를 만들어 매개변수로 받은 데이터들의 값들을 복사한다. 이는 MongoDB에서 \_id 라는 키 값을 하는 필드는 객체의 ID를 \_id로 매핑하기 때문에 전역변수로 선언한 data를 그대로 삽입하면 객체의 ID는 그대로이고 값들만 바귀기 때문에 한번만 삽입이 되고 나머지는 실패가 뜨기 때문이다.



<그림 x>

위의 <그림 x>는 insert\_data 객체에 값을 다 집어넣은 뒤, MongoDB 서버에 접속한 다음 특정 데이터베이스에 insert\_data를 집어넣는 코드이다.

ㅇ 특정 시간 스케쥴러 실행

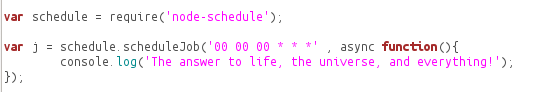
Node.js에서 특정한 날이나 반복적인 주기로 임의의 함수를 실행하려면 node-schedule이라는 모듈이 필요하다. node-schedule 는 npm install node-schedule로 설치가 가능하다.



<그림>

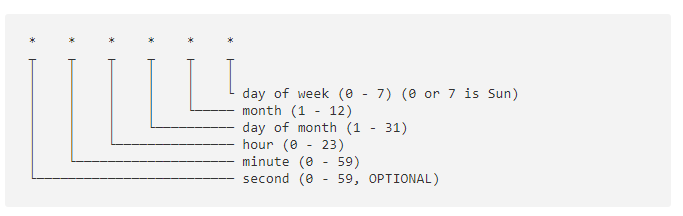
Node-schedule을 쓸 때 주의할 점이 몇가지 있는데 첫째로는 Node- Schedule는 interval-base가 아닌 time-base 스케줄링이다. 만약에 5분마다 한번씩 특정 함수를 수행시키기 위해서는 Node Schedule가 아닌 setInterval를 쓰는 것이 더 쉽고 효율적이다. 하지만 매달 3번째 주 화요일, 매 시간 :20분 , :50분에 특정한 함수를 수행하고 싶으면 Node Schedule가 적당한 라이브러리가 될 것이다.

두번째로는 Node Schedule는 in-process scheduling으로 설계되서 스크립트가 실행될 때만 동작하고 스크립트가 종료되면 스케쥴링도 동작하지 않는다. 만약에 스크립트가 종료되도 계속 실행되는 스케쥴러를 쓰고 싶으면 cron을 고려하는 것이 낫다.



<그림 x>

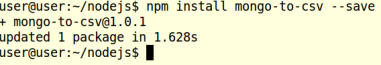
<그림 x>는 node-schedule 모듈을 이용해 자정마다 function() 함수가 실행되게 하는 코드이다. 이 코드를 설명하기 이전에 cron 형색 포맷을 이해해야 한다.



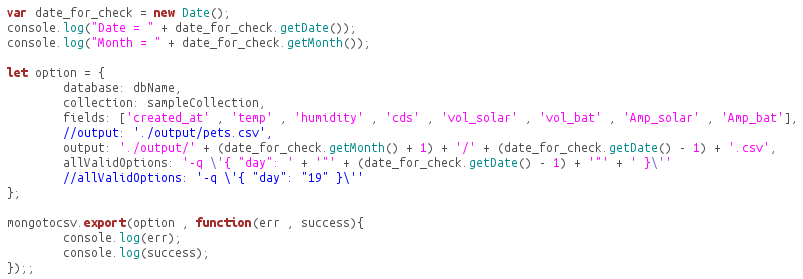
cron 형식의 포맷은 다음과 같이 구성된다. 본 서버에서는 cron 형식을 사용해서 날짜를 지정한다. 위의 포맷형식에 따라 00 00 00 \* \* \*은 매일 0시 0분 0초, 즉 자정마다 수행되는 함수이다.

ㅇ MongoDB를 csv 파일로 추출

npm은 csv 파일을 만들 수 있는 csv-stream 이라는 패키지를 제공하지만 MongoDB를 함수 하나로 csv 파일을 만들 수 있는 mongo-to-csv 라는 패키지가 존재한다. 본 연구에서는 그것을 이용하여 MongoDB에 있는 값들을 csv 파일로 저장한다. mongo-to-csv는 npm install mongo-to-csv -–save 명령어로 설치가 가능하다. 기본적인 설치법 , 설명 , API 함수는 <https://www.npmjs.com/package/mongo-to-csv> 를 참고한다.



<그림 x>

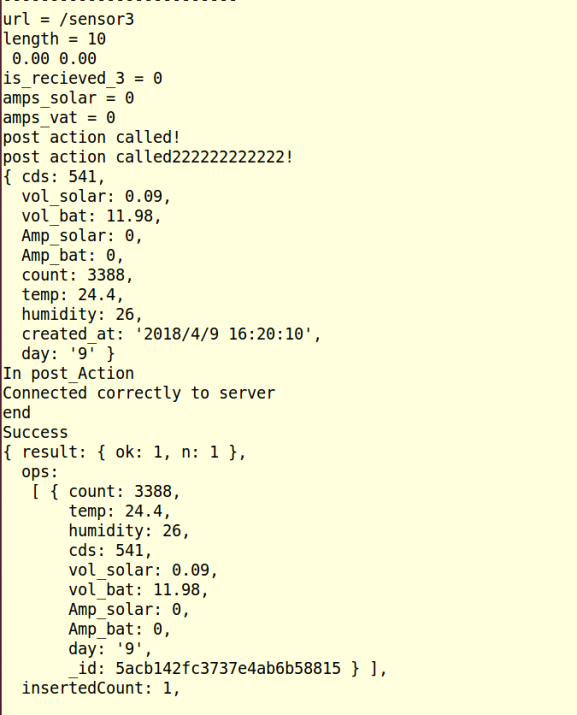


<그림 x>

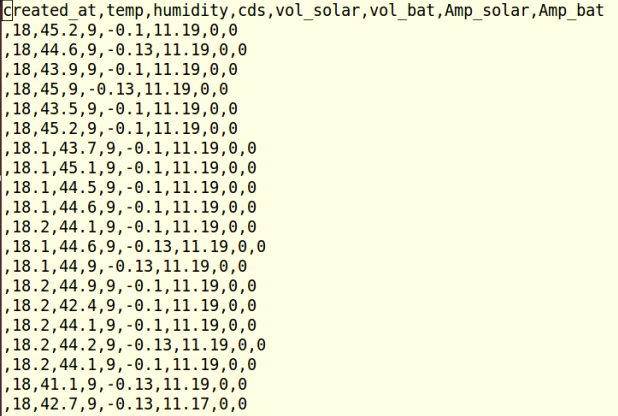
<그림 x>는 mongotocsv 함수를 이용하여 데이터베이스에 있는 값들을 csv 파일로 추출하는 코드이다. 먼저 현재의 시각을 구하여 월 , 일을 저장한 다음 option 변수를 선언해 옵션을 채운다. Database와 collection 속성으로 처음에 선언한 데이터베이스 이름과 콜렉션 이름을 넣고 fields 속성으로 4절에서 할 데이터 분석에 필요한 데이터들을 추출하고 output 속성으로 출력파일의 이름을 정한다.

allValidOptions 속성은 위의 mongo-to-csv 웹페이지에 가면 mongoexport 커멘드를 이용해서 옵션을 줄 수 있다고 한다. 본 서버에서는 DB에 들어가있는 day 필드를 이용해서 바로 전날의 데이터만 추출하는 옵션을 넣었다.

지금까지 설명한 기능을 수행하는 서버의 실행결과를 <그림 x>에 나타내었다.



<그림 x. 서버 실행 화면>



<그림 x. output 디렉토리의 csv 파일>

**4. 태양광 에너지 기반 IoT 데이터 분석**

‘IoT’와 ‘분석’은 현재 가장 각광받고 있는 IT트렌드에 속한다. 시장조사기관 가트너는 지난해 말 ‘2015년 10대 전략 기술 동향’을 발표, 향후 3년간 기업에 주요한 영향을 미칠 가능성이 있는 기술들 가운데 이 두 가지를 선정했다. IoT 시대를 맞아 디지털화로 인해 생성되는 데이터 흐름과 서비스의 융합은 관리(manage), 현금화(monetize), 운영(operate), 확장(extend)이라는 네 가지 IoT 사용 모델을 창조, 모든 기업들은 산업과 무관하게 이 기본 모델을 활용해 디지털 비즈니스를 영위할 수 있게 됐다.

아울러 임베디드(embedded) 시스템이 생성하는 데이터의 양이 증가하고 기업 내외 정형·비정형 데이터 풀(pool) 분석이 가능해지면서 분석이 보편화되고 있다. 기업들은 IoT, 소셜 미디어, 웨어러블 기기에서 생성된 대량의 데이터를 적절히 분류, 알맞은 정보를 제때 필요한 곳에 정확히 전달하는 것을 과제로 안게 됐다. 이에 가트너는 분석 기술이 모든 곳에 내장돼 끝단에서 데이터가 처리되는 ‘엣지 애널리틱스(Edge Analytics)’가 대두될 것으로 보고 있다.

R은 무료로 사용할 수 있으며, 다양한 패키지가 있고, 소스코드 또한 많으며 다양한 교육 재료가 있어 3절에서 구현한 센싱 디바이스로부터 얻은 데이터를 분석하는 데 적합하다.

본 연구에서는

1. 호주 The University Of Queensland에서 제공한 태양열 에너지 데이터를 참고하여 시간별 평균 에너지 생산량 출력
2. 호주 The University Of Queensland에서 제공한 태양열 에너지 데이터를 참고하여 각 환경변수 간의 상관계수분석
3. 3절에서 수집한 데이터를 참고하여 시간별 평균 에너지 생산량 출력
4. 3절에서 수집한 데이터를 참고하여 각 환경변수 간의 상관계수분석

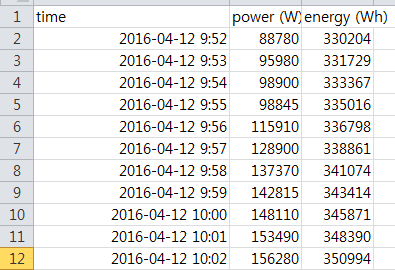
두 개의 결과를 통해 시간별 평균 에너지 그래프를 출력하고 상관계수를 분석하여 각 환경변수가 서로 얼마나 밀접한 연관관계를 가지는지 파악하고 본 연구가 공식데이터와 비교해서 얼마나 정확하게 나왔는지 분석한다.

■ 호주 The University Of Queensland에서 제공한 태양열 에너지 데이터를 참고하여 시간별 평균 에너지 생산량 출력

데이터 형식

호주에서 제공한 태양열 에너지 데이터는 전력량을 저장한 파일(pe20160412-20170412)이 필요하다.

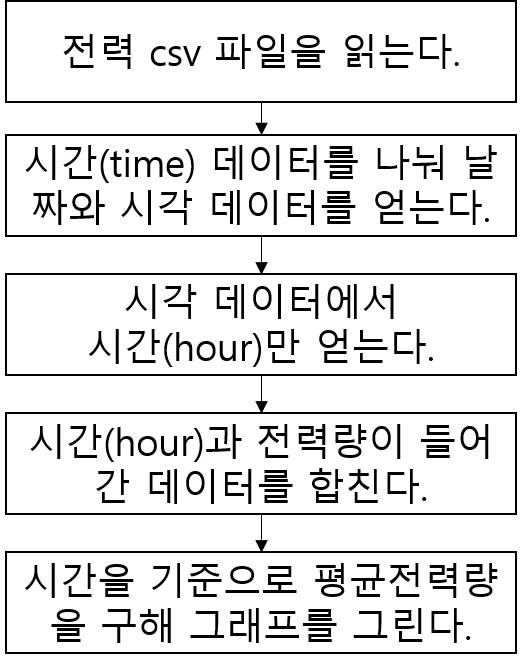
그리고 밑의 <그림 x>는 시간에 따른 전력량을 알 수 있는 표다.



<그림 x>

R 분석 알고리즘

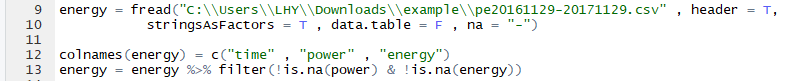
시간당 전력평균량을 구하는 기능을 수행하는 동작을 아래 <그림 x>에 순서도를 이용해서 나타내었다.



<그림 x>

R 분석 동작 상세 설명.

1. 호주에서 제공하는 데이터인 pe20161129-20171129.csv 를 읽어들이고 필드값에 변수 이름을 붙인 다음, 결손치가 있는 데이터는 제외한다.



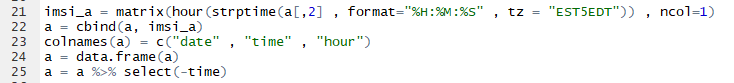
<그림 x>

1. 1.에서 읽어들인 데이터에서 time 필드인 2016-04-12 9:52 부분을 ‘ ‘를 토큰으로 해서 2016-04-12 와 9:52로 분리한다.



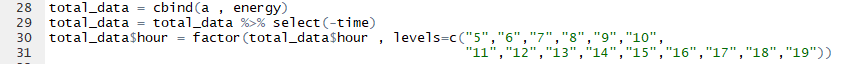
<그림 x>

1. 2.에서 분리해낸 9:52 부분의 시간 데이터를 시간(9) 부분만 얻어 필드값에 추가시킨다.



<그림 x>

1. 3.에서 얻은 시간(hour) 데이터를 전력량이 있는 원래 데이터와 합친다. 그러고 쓸모없는 필드인 시간(time) 데이터는 제거한다. 또한 hour 데이터는 숫자가 아닌 factor이므로 정렬해주지 않으면 그래프의 출력 순서가 이상하게 된다.



<그림 x>

1. 시간당 평균 전력량을 구하고 그래프로 출력한다.



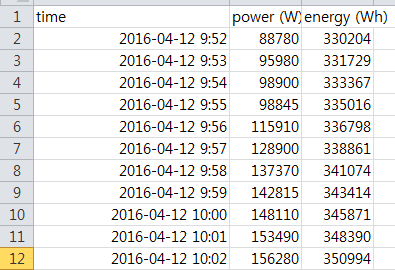
<그림 x>

■ 호주 The University Of Queensland에서 제공한 태양열 에너지 데이터를 참고하여 각 환경변수 간의 상관계수분석

데이터 형식

호주에서 제공한 태양열 에너지 데이터는 전력량을 저장한 파일(pe20160412-20170412) , 기후정보를 저장한 파일(w20160412-20170412)이 필요하다.

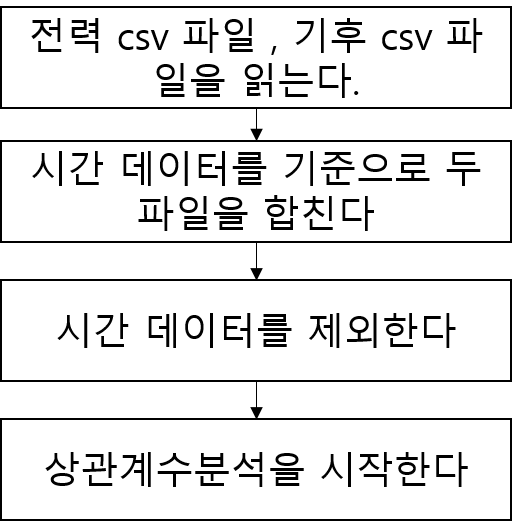
그리고 밑의 <그림 x>는 시간에 따른 전력량을 알 수 있는 표다.



<그림 x>

R 분석 알고리즘

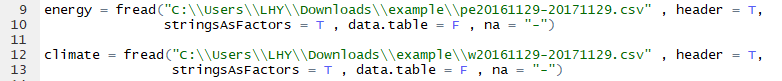
시간당 전력평균량을 구하는 기능을 수행하는 동작을 아래 <그림 x>에 순서도를 이용해서 나타내었다.



<그림 x>

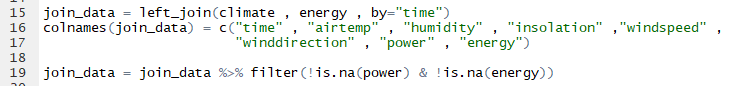
R 분석 동작 상세 설명.

1. 호주에서 제공하는 데이터인 pe20161129-20171129.csv , w20161129-20171129.csv 를 읽어들인다.

****

<그림 x>

1. 1.에서 읽어들인 데이터 2개를 시간에 대하여 합친다. 또한 필드값에 대하여 변수값을 지정한다.



<그림 x>

1. 시간데이터는 숫자가 아니기 때문에 시간데이터가 있는 데이터프레임을 상관계수분석 함수에 집어넣으면 오류가 뜬다. 따라서 시간데이터는 제외하고 상관계수분석을 수행한다.



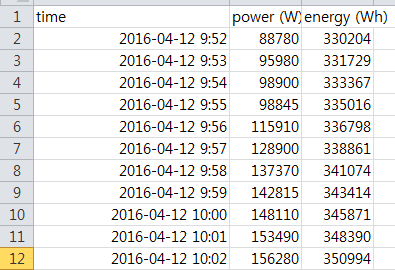
<그림 x>

■ 호주 The University Of Queensland에서 제공한 태양열 에너지 데이터를 참고하여 월별 평균 전력생산량 그래프 그리기

데이터 형식

호주에서 제공한 태양열 에너지 데이터는 전력량을 저장한 파일(pe20160412-20170412) , 기후정보를 저장한 파일(w20160412-20170412)이 필요하다.

그리고 밑의 <그림 x>는 시간에 따른 전력량을 알 수 있는 표다.



<그림 x>

R 분석 알고리즘

대략적인 알고리즘은 제일 처음에 구해본 호주 The University Of Queensland에서 제공한 태양열 에너지 데이터를 참고하여 시간별 평균 에너지 생산량 출력과 비슷하다.



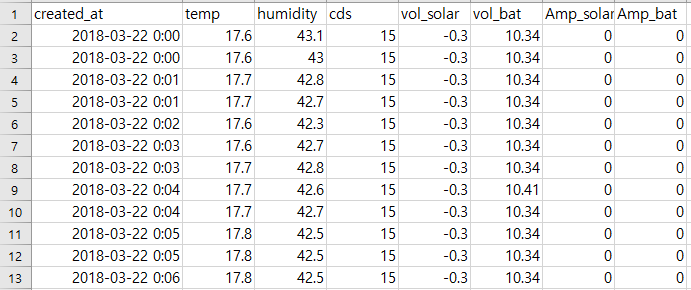
<그림 x>

위 <그림 x>는 전체적인 코드이다. 시간별 평균량은 19줄의 코드에서 hour 함수를 통해 시간을 구하는 거였지만 위의 그림에서는 month로 달을 구한다.

■ 3절에서 수집한 데이터를 참고하여 시간별 평균 에너지 생산량 출력

데이터 형식

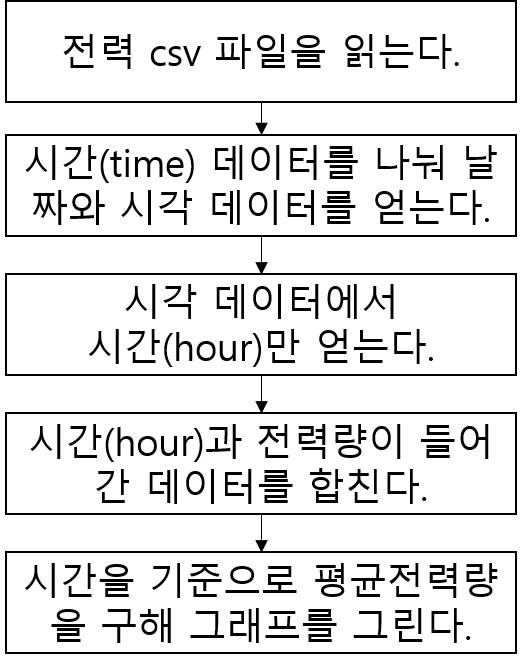
3절에서 아두이노로부터 받은 데이터를 csv 파일로 저장한 output/(월)/(일).csv 파일이 필요하다. 데이터 형식은 밑의 그림과 같다.



<그림 x>

R 분석 알고리즘

시간당 전력평균량을 구하는 기능을 수행하는 동작을 아래 <그림 x>에 순서도를 이용해서 나타내었다.



<그림 x>

R 분석 동작 상세 설명.

1. 3.2절의 Node.js가 제공하는 output/3/22.csv 를 읽어들이고 전류값 , 배터리 전압은 제외시킨다.



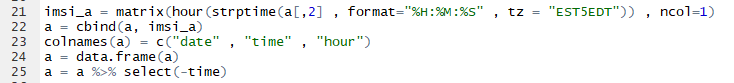
<그림 x>

1. 1.에서 읽어들인 데이터에서 time 필드인 2018-03-22 0:00 부분을 ‘ ‘를 토큰으로 해서 2018-03-22 와 0:00로 분리한다.



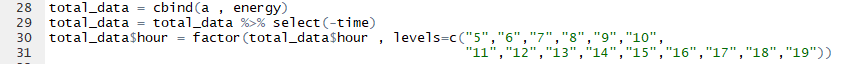
<그림 x>

1. 2.에서 분리해낸 9:52 부분의 시간 데이터를 시간(0) 부분만 얻어 필드값에 추가시킨다.



<그림 x>

1. 3.에서 얻은 시간(hour) 데이터를 전력량이 있는 원래 데이터와 합친다. 그러고 쓸모없는 필드인 시간(time) 데이터는 제거한다. 또한 hour 데이터는 숫자가 아닌 factor이므로 정렬해주지 않으면 그래프의 출력 순서가 이상하게 된다.



<그림 x>

1. 시간당 평균 전력량을 구하고 그래프로 출력한다.

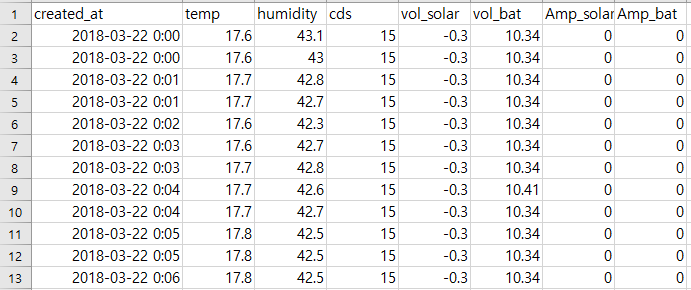


<그림 x>

■ 3절에서 수집한 데이터를 참고하여 각 환경변수 간의 상관계수분석

데이터 형식

3절에서 아두이노로부터 받은 데이터를 csv 파일로 저장한 output/(월)/(일).csv 파일이 필요하다. 데이터 형식은 밑의 그림과 같다.



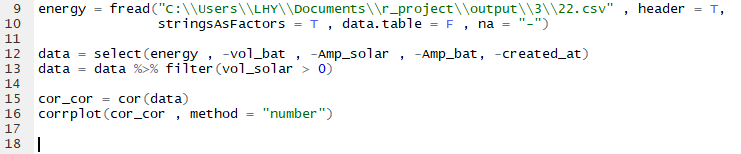
<그림 x>

R 분석 알고리즘

다른 분석과는 달리 Node.js로부터 추출한 csv 파일은 시간 , 기후 , 전압 데이터가 다 나와있어서 데이터를 합치지 않아도 되므로 간단하다.

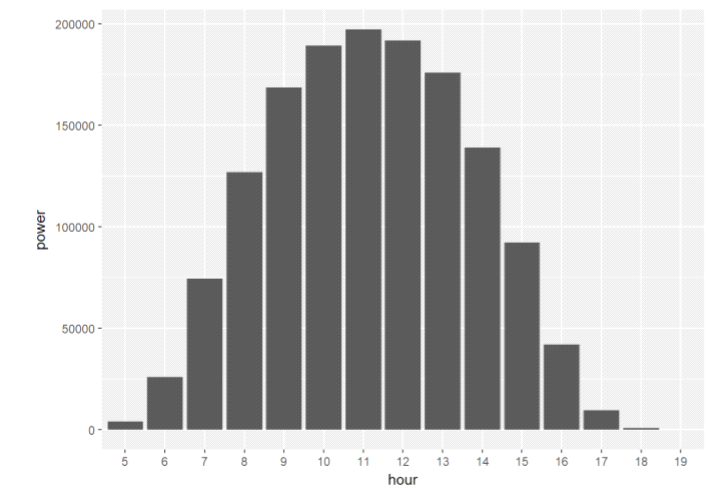
단지 파일을 읽어서 상관계수분석이 불가능하거나(created\_at) , 할 필요가 없는 필드들(vol\_bat , Amp\_solar , Amp\_bat) 제거한 후에 상관계수분석을 수행하면 된다.

R 분석 동작 상세 설명.

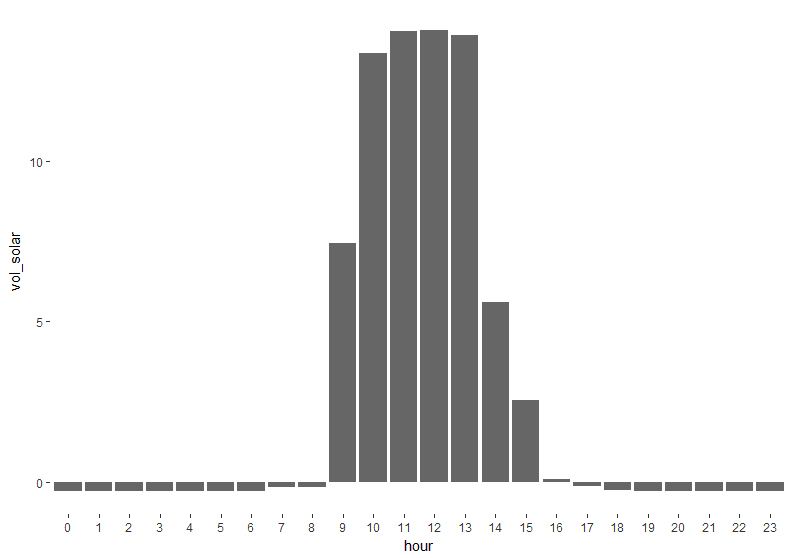


<그림 x>

밑의 <그림 x> 는 호주에서 받은 데이터와 IoT 센싱 디바이스로부터 얻은 데이터로 시간당 평균 전력에너지 , 전압값을 그래프로 나타낸 것이다.



<그림 x. 호주 데이터로 시간당 평균 전력량>



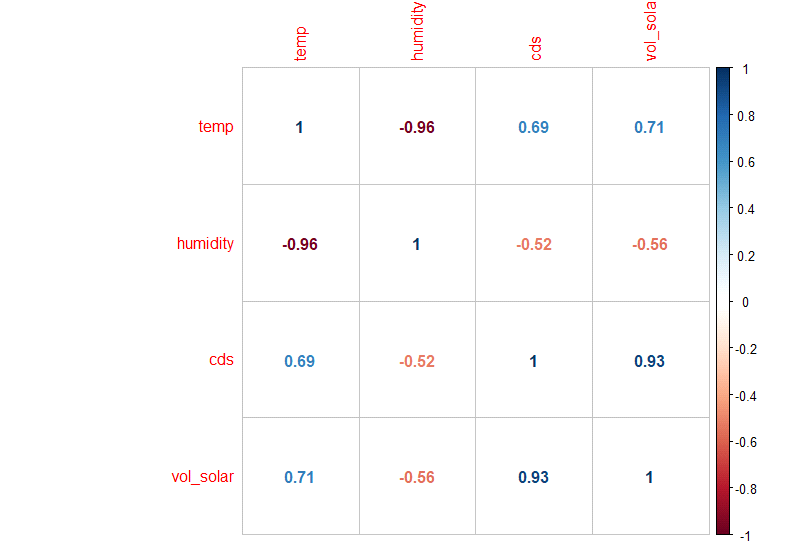
<그림 x. IoT 센싱 디바이스로 수집한 데이터의 시간당 평균 전압량>

시간당 에너지 생산량 그래프에서는 꽤나 차이가 나타나는데 이는 본 실험에서 데이터를 얻는 것은 실내에서 행해졌기 때문에 9~3시까지만 실내에 태양빛이 들어오기 때문이다. 그것을 제외하고 생각하면 대략적인 그래프 분포는 비슷하게 나오게 된다.

밑의 <그림 x>는 본 실험에서 생산해낸 IoT 센서 디바이스로 측정한 데이터로 상관계수분석, 호주에서 받은 데이터로 상관계수분석을 실시한 결과이다.



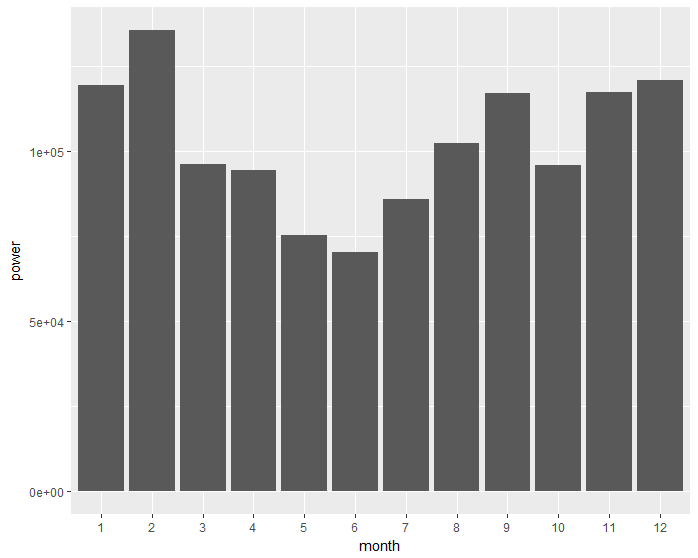
<그림 x. 호주 데이터의 환경변수로 상관계수분석 결과>



<그림 x. IoT 센싱 디바이스로 측정한 변수로 상관계수분석 결과>

위의 시간당 평균 생산량 그래프와는 다르게 상관관계분석은 흡사하게 나온다. 위의 호주의 상관계수분석에서는 전력과 일사량의 상관계수는 0.9, 온도는 0.44, 습도는 -0.52이다. 그에 반해 IoT 센서로 수집한 데이터로 측정한 상관계수는 온도가 0.71, 습도가 -0.56, 조도가 0.93이다.

밑의 <그림 x>는 월별 평균 전력생산량을 그래프로 나타낸 것이다. 호주는 남반구라 여름에 제일 전력생산량이 작고 겨울에 전력생산량이 크게 나온다. 본 연구에서는 데이터가 부족해 IoT 센싱 디바이스로 받은 데이터로는 월 별 평균생산량을 구하기가 불가능했다.



<그림 x. 월별 평균생산량>