**졸업 프로젝트**

**테스트 결과서**

-아두이노와 웹서버를 이용한 스마트홈 오토메이션-

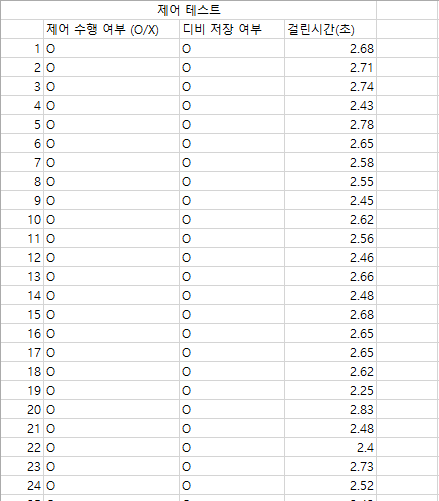
고지혜 201311193

김예찬 201311200

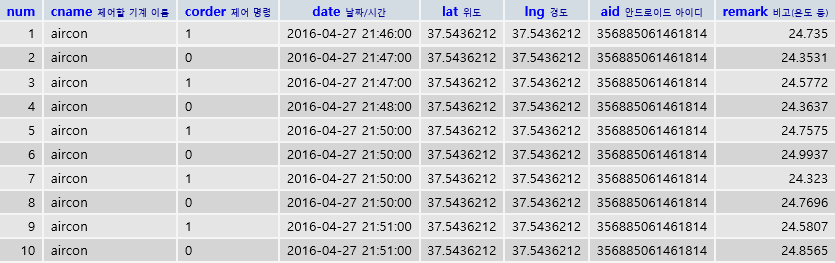
1. **사물 제어**

사물 제어에 대한 테스트는 사물 제어 명령을 100개를 보냈을 때, 몇 개가 제대로 수행되는지 얼마나 걸리는 지에 대해서 테스트를 함으로서 기능적 요구사항과 이와 관련된 비기능적 요구사항을 동시에 테스트 하였다.

테스트 결과는 아래와 같다. 결과 기록은 사용자가 제어 버튼을 눌렀을 때로부터 몇 초 후에 사물이 제어되는지, 그리고 그 사물의 변경된 상태가 서버 데이터베이스에 저장이 되는지에 대한 여부이다. 아래는 그 결과를 표로 나타낸 것이며 전체 테스트 결과는 파일로 첨부하였다.



< 100회의 수행에 따른 결과 표>



< 100회의 수행에 따른 데이터베이스 삽입 여부>

이 기능과 관련된 비기능적 요구사항으로는 성능 부분에서 ‘제어시간은 **5초** 이내여야 한다. ‘였고, 이에 맞게 수행되었는지 확인한 100개의 수행의 걸린 시간에 대한 그래프는 아래와 같다.

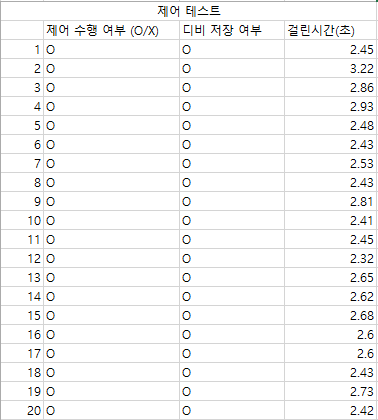
< 100회의 수행에 따른 수행시간 결과 그래프 >

100개의 테스트에 대해서는 100%로 모두 사물이 정상적으로 수행되었으며, 수행 시간의 평균은 **2.63**초로 나왔고, 최고치도 3초를 넘지 않았다.

1. **거리에 따른 제어**

사물 제어 거리에 대한 테스트는 일반 가정의 집 크기를 고려하여, 서브 아두이노를 중앙 아두이노로부터 1M, 5M, 10M 거리에 위치시킨 후 제대로 수행되는지 그리고 그 수행이 얼마나 걸리는 지에 대해서 테스트 하였다.

테스트 결과는 아래와 같다. 결과 기록은 제어 후 상태가 바뀌는 데 몇초가 걸렸는지와 디비 저장 여부를 기록하였다. 모든 기록은 파일로 첨부하였다.



< 100회의 수행에 따른 결과 표>

수행은 각 단위(1M, 5M, 10M)에 대해서 20회씩 수행하였으며, 제어와 디비 저장은 100%의 확률로 정상적으로 수행하였다. 그리고 총 60회에 따른 결과 그래프는 아래와 같다.

< 거리별 20회(총 60회)의 수행에 따른 결과 표>

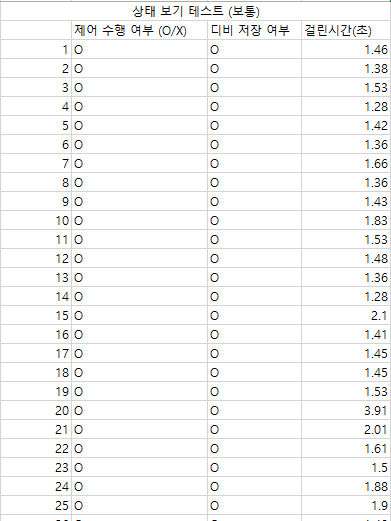
결과는 3개의 거리 차이에 대해서 크게 차이가 없었다는 것을 알 수 있다. 또한 제어 시간이 거리에 상관 없이도 99%의 확률로 비기능적 요구사항을 지켰다는 것을 확인할 수 있었다. 결과로 나온 평균 소요시간은 1M 일 때 **2.60초**, 5M 일 때 **2.61초**, 10M 일 때 **2.67초**만큼 소요되었다.

1. **사물 상태 확인**

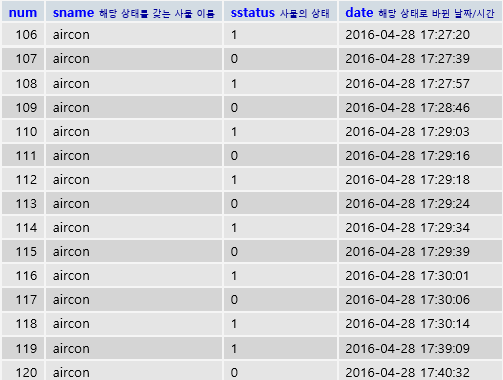
사물의 상태를 확인하는 테스트는 사물의 상태를 서버 혹은 서브 아두이노로부터 사용자의 스마트 폰에 이르기 까지 어느정도의 시간이 걸리는 지에 대해서 테스트를 하였고, 그 상태가 데이터베이스에 제대로 저장되는지에 대해서 확인하면서 기능, 비기능적 요구사항을 테스트하였다.

* 1. **사물 상태 정보 확인(보통 상태)**

온도가 아닌 제어할 수 있는 사물의 상태를 받아오는 것은 실시간성을 요하는 정보가 아니기 때문에 데이터베이스에 저장되어 있고, 테스트로는 사용자의 스마트폰에서 사물의 상태 요청을 했을 때 사용자의 스마트 폰에서 확인하는 데에 걸리는 시간을 계산하는 것이며 그 결과는 아래와 같이 나왔다.



< 100회의 수행에 따른 결과 표>



< 100회의 수행에 따른 데이터베이스 삽입 여부>

이 수행도 마찬가지로 100회의 수행 모두 정상적으로 사물의 상태를 불러왔고, 그 상태가 디비에 제대로 인서트되어 있었다. 이에 대한 결과 파일은 첨부하였다. 아래는 100회 수행에 따른 수행시간 결과 그래프이다.

< 100회의 수행에 따른 결과 표>

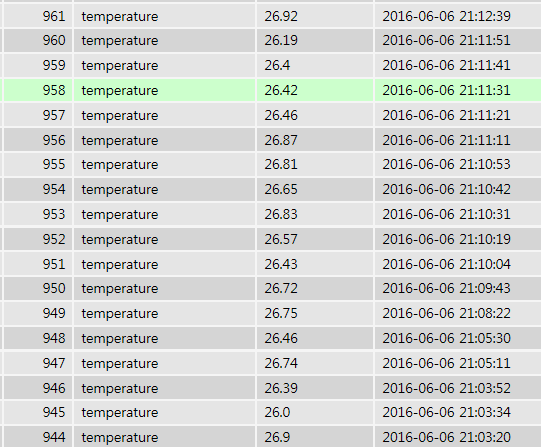
이 결과는 다른 상태(온도)와 차이가 나는 것은 이는 온도처럼 서브 아두이노까지 요청을 할 필요 없이 서버 데이터베이스에 접근하여 정보를 받아오기만 하면 되는 것이다. 그래서 온도 상태를 받아오는 것과 시간적으로 차이가 났다. 결과로 나온 평균 소요 시간은 **1.64**초 였다.

* 1. **실시간 상태 정보 확인 (온도)**

실시간 상태 정보(온도)를 확인하는 것에 대한 테스트는 서브 아두이노에서 사용자의 스마트 폰에서 확인하는 데에 걸리는 시간을 계산하는 것이며 그 결과는 아래와 같이 나왔다.



<수행에 따른 결과 표>



< 수행에 따른 데이터베이스 삽입 여부 (일부 캡쳐)>

30회의 수행에 대한 결과로는 98%의 확률로 온도를 사용자의 스마트폰에서 실시간으로 확인할 수 있었고, 디비에도 동일 확률로 인서트되었다. 이에 대한 결과 파일은 첨부하였다. 아래는 30회 수행에 따른 수행시간 결과 그래프이다.

< 100회의 수행에 따른 수행시간 결과 그래프>

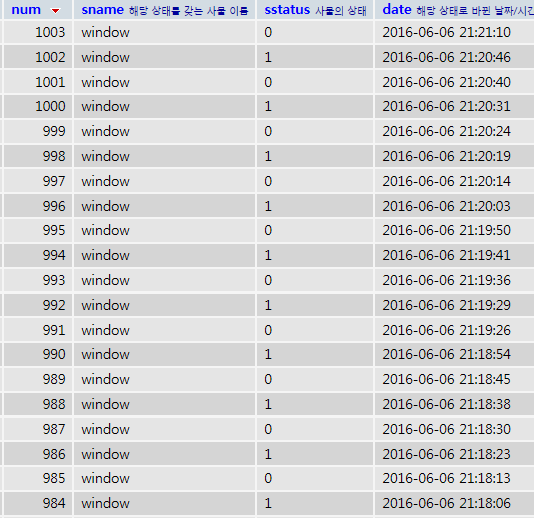
처음 실시간 상태 보기를 구현하였을 때에는 6~7초의 결과가 나왔다. 이 후 시간을 줄이기 위해 다른 방식으로 구현하여 시간이 4~5초로 줄었다. 사용자의 안드로이드 기기에서 서버로 요청을 하고 서버는 다시 아두이노로 요청을 한 후에 온도를 받아오고 마지막으로 사용자에게 전달되는 라운드 트립 통신임은 변함이 없기에 보통 상태 보기보다는 시간이 오래 걸렸다. 평균 소요시간은 평균 **4.89초** 였다.

* 1. **긴급 상태 정보 확인 (창문)**

긴급 상태 정보(창문)를 확인하는 것에 대한 테스트는 창문이 열렸을 때 사용자의 안드로이드 앱에 푸시 알림이 오기까지 걸리는 시간을 계산하는 것이며 그 결과는 아래와 같이 나왔다.



<수행에 따른 결과 표>



< 수행에 따른 데이터베이스 삽입 여부 (일부 캡쳐)>

30회의 수행에 대한 결과로는 100%의 확률로 창문이 열렸다는 푸시알림이 사용자의 안드로이드 기기로 왔고, 데이터베이스에도 동일 확률로 인서트 되었다. 이에 대한 결과 파일은 첨부하였다. 아래는 30회 수행에 따른 수행시간 결과 그래프이다.

< 30회의 수행에 따른 수행시간 결과 그래프>

GCM을 통해 푸시 알림이 오는 속도는 웹서버가 GCM서버로 메시지를 보낸 후 그 메시지가 안드로이드 기기로 오는 것이므로 역시 보통 상태 보기에 걸리는 시간보다는 좀 더 시간이 소요되었다. 평균 소요시간은 평균 **4.37초**였다.

상태보기 기능과 관련된 비기능적 요구사항으로는 성능 부분에서 ‘상태보기시간은 **5초** 이내여야 한다. ‘였고, 99%의 확률로 해당 비기능적 요구사항이 지켜진 것을 확인할 수 있다.

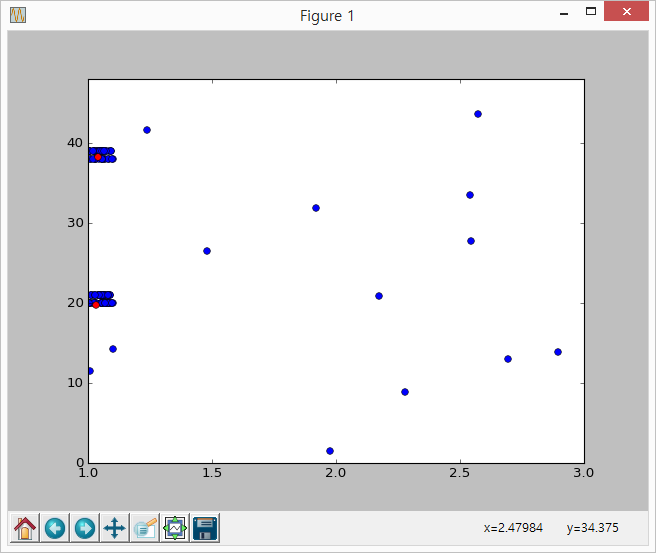
1. **패턴**

K-Medians 알고리즘의 정확도에 대하여 테스트하였다. 사용자의 제어 패턴을 분석할 때 몇몇의 outlier들은 당연히 생기기 마련이고, 이에 대해서 알고리즘을 수행해보았다.

입력으로는 120개의 제어 기록을 사용자의 두 가지 패턴에 대해서 모이게 하고, 그 중 10%에 대해서는 random하게 입력으로 주어 outlier(만약 이게 outlier가 아닌 한곳에 뭉쳐 있었다면 Elbow Method에 의해서 새로운 클러스터가 생성되었을 것임) 에 얼마나 민감한지를 확인하려고 한다. 결과는 아래와 같고, 각 축의 의미는 다음과 같다.

x축: 제어했을 때의 사용자의 집까지의 거리(사용자의 위치)

y축: 제어했을 때의 시간(30분 간격으로 나눔)



10%의 outlier에도 사용자에게 패턴서비스의 값으로 제공되는 클러스터의 중심 값은 중간 값을 선택하기 때문에 데이터가 많이 보여져 있는 사용자의 패턴 부분에 속해있음을 볼 수 있다.