|  |
| --- |
| 안녕하세요 스마트홈 발표를 맡은 고지혜 입니다. 지금부터 발표를 시작하겠습니다. |
| 발표 순서는 다음과 같습니다. 먼저 간략하게 저희 프로젝트 목적을 다시 설명드리고, 저번 발표 이후로 구현한 내용에 대해 설명하겠습니다. 그 후 일부분의 요구사항에 대한 시험 계획 및 결과에 대해 말씀드리고 향후 일정을 말씀드리겠습니다. |
| 프로젝트 목적입니다. |
| 저희 프로젝트는 다들 아시다시피 안드로이드 앱으로 집안의 사물들을 제어할 수 있는 홈 오토메이션 시스템 개발하는 것 입니다. |
| 다음은 구현 내용입니다ㅣ. |
| 먼저 메시지 프로토콜입니다. 저번 발표 때 개선할 사항에서 말씀드렸다 시피 웹소켓을 통해 주고받는 메시지가 많아서 메시지 프로토콜이 필요합니다.  먼저 사용자가 사물 제어 명령을 내릴 때는 “sendControl/cname/corder”의 형식을 가집니다. cname – 사물의 이름, corder – 명령의 내용입니다. 이 메시지는 중앙 아두이노를 통해 서브 아두이노로 전달되어 사물을 제어하게됩니다. 두번째로 상태 요청인데요, 이 경우는 두가지로 나뉩니다. 에어컨, 조명과 같이 On/Off만 존재하는 보통의 상태의 경우 아두이노쪽과 관계없이 DB에 최근 저장된 상태를 가져옵니다. 하지만 온도의 경우 실시간성이 중요하기 때문에 상태 요청 메시지를 아두이노로 보냅니다. 메시지 형태는 “requestStatus/sname”의 형태입니다. 이 메시지를 보내고 웹서버는 상태 정보 메시지를 기다리게 됩니다.  아두이노가 보내는 상태정보는 세가지가 있는데요, 에어컨이나 조명의 On/Off 여부인 보통의 상태, 실시간성이 중요한 온도, 창문의 여닫힘 여부인 긴급성이 있는 상태가 있습니다. 각각 포맷은 이렇습니다. normalStatus와 emergency status는 아두이노는 상태가 바뀔 떄마다 서버로 메시지를 보내고, realtimeStatus는 상태 요청이 들어오면 그 응답으로 메시지를 보냅니다. 서버는 normalStatus가 들어오면 단순히 DB에 상태를 저장합니다. realtimeStatus의 경우에도 DB에 저장을 한 후, 서버가 상태요청을 하고 기다리고 있는 상태이기 때문에 해당 플래그값을 바꿔서 바로 XML형태로 출력하여 안드로이드에 응답을 보냅니다. EmergencyStatus의 경우엔 사용자가 상태 정보 보기를 안해도 그 상태를 알려야하기 때문에 GCM을 통해 사용자에게 푸시알림을 보냅니다. |
| 다음은 상태 확인을 인터페이스를 구현한 부분입니다. 이전 상태에 더하여 온도에 대한 정보를 가져오는 부분을 추가했고, 가져오는 방식은 이전 발표에서 언급했듯, 스레드를 이용해서 xml 파싱을 해서 가져온다! 그리고 온도 같은 경우에는 제어할 때 볼 수 있으면 더 편리하므로 제어를 하는 액티비티에서도 확인할 수 있도록 추가하였습니다. |
| 다음은 데이터베이스 설계 부분입니다. 패턴 인식 알고리즘을 돌리기 위해 Pattern table을 추가하였습니다. Time은 24시간을 30분으로 나눈 값입니다. 기존에는 사용자의 명령들을 저장하는 Control테이블엔 ~~필드들이 있었습니다. 하지만 사용자의 패턴을 분석하려면 위의 기본적인 필드들 외에 부가적인 정보가 필요했습니다. 사용자가 어떤 온도일 때 에어컨을 제어했는지의 정보가 필요합니다. 따라서 remark라는 비고 필드를 추가하였습니다. |
| 마지막으로 패턴 분석 알고리즘 부분입니다. 이전 발표에서 말씀드렸다시피 두 개의 알고리즘을 비교한 후 선택하기로 하였는데요, 비교는 두 부분(패턴이라고 생각되는)에 값을 집중시키고, outlier데이터를 랜덤으로 넣었을 때 클러스터의 중심이 어떻게 변하는지에 대해 알아보는 것이엿다. 결과는 두개의 그래프에서 확인할 수 있고, K-Means는 클러스터의 중심을 평균값을 선택하기 때문에 어느정도의 outlier에도 민감하게 반응을 하고, K-Medians는 중심을 중간값으로 선택하기 때문에 덜 민감하게 반응하였다. 그래서 덜민감한 K-Medians를 구현하여 사용하기로 결정하였습니다. |
| 다음은 시험 계획 및 결과입니다. |
| 저번에 피드백 받은대로 패턴 분석 알고리즘의 시험 계획을 고쳤습니다. 이 시험은 outlier가 패턴에 영향을 줄 때 전체 데이터에서 어느 정도의 비율을 갖는 지 확인하는 시험입니다. 그래서 기존 패턴이 들어가 있는 데이터에 outlier를 조금씩 추가하면서 패턴의 값이 어떻게 변하는 지 확인합니다. 또한 이렇게 나온 패턴이 제대로 사물제어가 되는지 확인합니다. |
| K-means와 k-median를 비교하고 알고리즘을 구현하느라 아직 시험을 수행하진 못하였습니다. 다음 발표 때 보여드리도록 하겠습니다. |
| 다음은 성능 테스트입니다. 테스트 한 결과 정해놓았던 시간보다 더 짧은 시간이 걸린다는 것을 알게 되어 사물 제어 시간을 줄였습니다. 반면에 실시간 사물 상태를 불러오는 시간은 사용자가 상태를 요청하고, 아두이노로 그 메시지를 보내고, 다시 상태 값을 받아오는 라운드 트립이기 때문에 시간이 길었습니다. 시험방법은 저번처럼 제어와 상태데이터를 100개씩 보내어 확인하는 것입니다. 몇 개가 제대로 수행되는지 얼마나 걸리는 지에 대해서 테스트를 함으로서 기능적 요구사항과 이와 관련된 비기능적 요구사항을 동시에 테스트 하였습니다. |
| 먼저 제어명령의 테스트 결과입니다. 수행은 100프로 다 함 100%의 확률로 3초를 넘지 않는다. 평균 소요 시간은 2.63초 입니다. |
| 다음은 보통의 상태보기의 테스트 결과입니다. 100프로 상태가 저장이 되고, 제대로 출력이 되었고, 98퍼센트가 3초를 넘지 않았습니다. 네트워크상황때문에 간혹 3초를 넘는 것으로 보입니다. 하지만 평균값은 1.64초로 오히려 제어시간보다 짧습니다. |
| 마지막으로 실시간의 상태보기의 테스트 결과입니다. 100프로 상태가 저장이 되고, 제대로 출력이 되었고, 95퍼센트가 7초를 넘지 않았습니다. 평균 소요시간은 6.56초로 꽤 깁니다. 라운드 트립인만큼 스마트폰의 네트워크 상황, 웹서버의 네트워크 상황등 결정 요인이 많기 때문에 시간을 초과할 때가 많은 것으로 보입니다.  마지막으로 향후일정입니다. |
| 앞서 말씀드린 것처럼 긴급 상태의 경우 GCM을 통해 사용자에게 푸시알림을 보내는 기능을 구현할 예정입니다. 그리고 간혹 웹소켓이 끊기는 문제에 대해 알아보고 수정할 예정입니다. 그리고 새로운 데이터가 들어왔을 때 오버헤드가 커지는 경우에 대해서 알고리즘을 어떻게 수행할 것인지에 대해서 수정하고 테스트할 예정입니다. 그리고 최종적으로 마무리 하고 다음 발표 때 시연하도록 하겠습니다. |
| Thankyou |