임베디드 소프트웨어 경진대회 The World Embedded Software Contest

제17회 임베디드SW경진대회

개발완료보고서 [스마트 홈]

산업통상자원부

KESSIA 임베디드소프트웨어·시스템산업협회 Korea Embedded Software and System Industry Association

0. 작성 시 주의사항

※아래의 작성 양식(제출분량, 폰트, 크기, 줄 간격 등)을 미준수 시 서류 평가의 감점요인됨

※ 제출 분량 : A4 용지 상세내용 포함 30 page 이내

※ 작성 양식 (폰트 : 맑은 고딕 / 폰트 크기 : 10pt / 자간 : 0% / 장평 : 100% / 줄 간격 : 130%)

※ 제출 포맷 : pdf

1. 팀 정보

팀명	믿똘팀	팀장	조성익
팀원	성기정	팀원	김유현
팀원	김영평	팀원	정현우

2. 개발완료보고서

0. 작품명 : 홈케어, 어떡케어

1. 개요

1.1. 작품 개요

한국무역협회 국제무역연구원의 보고서에 의하면, 스마트홈 관련 시장 규모가 전 세계 중 1위인 미국의 트렌드는 인공지능, 보안, 호환성, 가격 등인 것으로 나타났다. 자가 설치가 가능하고 보유한 제품들과 상호 호환할 수 있는 제품 등 서비스를 구축하는 데 있어 상대적으로 저렴한 제품들이 사랑을 받았다고 한다. 실제로 음성인식과 AI 비서를 통한 스마트 스피커가 빠른 속도로 가정에 도입되고 있으며, 도난 방지 및 범죄 예방을 목적으로 하는 가정 보안 관련 제품도 빠르게 성장하고 있다. [1]

[1] 세계 1위 스마트홈 시장 미국의 트렌드는 '보완, 호환성'(아시아경제, 2018. 11. 11)

스마트홈 시장 1위 미국, 보안 서비스 및 상호운용성 중시(산업일보, 2018. 11. 13)

음성인식은, 한국에서는 여전히 많은 부분에서 아쉬움을 보인다. 발음의 유사성 (이를테면, '으'와 '의' 등)을 제대로 구분하지 못하는 경우가 많고, 사투리의 단어나 억양이 포함되어 있다면 오인식률은 더 상승한다. 우리 팀의 작품은 나머지 세 개의 트렌드를 겨냥했고 특히, 자가 설치와 상호 호환성 부분에 큰 강점이 있도록 했다. 기본적인 역할을 수행하고 있는 상태에서 TV(라즈베리파이)를 통해 상태를 변화시킬 수 있도록 하였다. 예를 들어, 사용자가 원한다면 작동을 멈출 수도 있고 다시 실행시킬 수도 있다.

우리의 스마트홈은 하나가 아닌 여러 개의 모듈로 존재한다. 입출입을 알 수 있는 현관과 날씨를 알려주고 우산을 건조시키는 우산꽂이, 신발을 관리하는 신발장, 사용자의 칫솔을 보관하고 사용횟수를 알려주는 칫솔걸이, 습기를 자동으로 조절하는 욕실, 그리고 이 모든 것을 확인할 수 있는 스피커와 WebOS기반의 대쉬보드. 각자 따로 움직이며 다른 역할을 수행하지만 하나의 서버에 연결되어 있고 서로의 데이터를 주고받는다. 각 작품이 단순하지만 명확한 기능을 수행하기 때문에 별다른 지시가 없어도 각자의 임무를 수행할 수 있다. 사용자는 TV(라즈베리파이)를 키지 않아도 스마트한 혜택을 누릴 수 있다. 생각해보면, 사용자의 명령을 기다려야 한다는 것은 'Smart'하지 않다. 사용자의 명령은 절대적이어야 하지만 동시에 부가적인 것이어야 한다.

우리는 이러한 제품이 많아져야 스마트홈이 더욱 성장할 것이라고 생각한다. 현재 스마트홈이 우리에게 친숙한 이미지는 아니다. 친숙 해지려면 좀 더 우리의 일상 안으로 침투해야 한다고 생각한다. 사용자는 나만의 스마트 홈을 만들 수 있어야 하고, 사용자에게 귀찮게 묻지 않아도 각 가전이 사용자가 원하는 역할을 알아서 수행할 수 있으며, 또한 사용자가 원한다면 현 상황파악과 제어를 바로 할 수 있을 때 'Smart Home'이 된다고 생각한다. 마치 우리가 스마트 폰을 다양한 어플리케이션들로 원하는 대로 꾸미는 것처럼 말이다.

우리는 스마트 홈의 초안을 제안한다. 아주 기본적인(즉, 저가의) 센서와 방법으로 기능을 수행하도록 했다. 우리가 가진 능력과 자본이 제한적이었기 때문이다. 그렇기 때문에 우리의 제안은 더 좋은 센서, 더 좋은 기술을 사용한다면 더 강력한 퍼포먼스와 더 좋은 편의를 제공할 수 있다는 가능성을 포함하고 있다.

1.2. 개발 목표

자가설치가 가능하고 쉽게 교체할 수 있으며 사용자를 귀찮게 하지 않는 제품. 모든 제품은 사용자가 원한다면 확인할 수 있고 명령을 내려 제어(Control)할 수 있는 제품. 완성이 아니라 시간이 지남에 따라 더더욱 발전할 가능성을 포함하고 있는, 그런 초석이 되는 제품을 개발하는 것이 목표이다.

2. 개발 환경 설명 (최대한 자세하게 기술)

2.1. Hardware 구성

각 하드웨어에 번호를 붙였다. (표 1) 번호를 붙인 이유는 우리의 하드웨어는 모두 클라이언트이며 이 클라이언트 모두가 서버에 연결되어 있고 클라이언트가 데이터를 전송할 때 서버를 거치게 된다. 모든 클라이언트가 같은 포트를 사용하고, 전송되는 데이터는 문자열이다. 이 방법의 문제는 각 하드웨어마다 겹치는 명령이 있을 수 있다는 것이다. 예를 들면, 신발장과 우산꽂이, 욕실 모두 'FANON'과 'FANOFF' 명령을 사용한다. 그러므로 데이터명을 각 파트마다 확실하게 구분할 수 있지만 그 데이터의 의미가 무엇인지를 분명하게 알 수 있는 데이터명으로 정해야 했다.

그래서 각 하드웨어마다 번호를 붙이고 명령어마다 번호를 붙이게 해서 혼란을 막도록 했다. 표 1을 참조해서 신발장, 우산꽂이의 FANON 명령을 만들어보자면 신발장은 '3FANON'이 되고 우산꽂이는 '4FANON'이 된다. 후설 하겠지만, 이 명령어는 단순한 예시이며 실제로 사용한 변수명은 다르다.

번호	**하드웨어**
1	현관
2	스피커
3	신발장
4	우산꽂이
5	칫솔걸이
6	욕실

표 1. 하드웨어의 번호

1) 현관

현관 시스템은 사용자가 집에서 처음으로 만나면서 동시에 마지막으로 만나는 '스마트 가전'이다. 본 작품에서는 두 개의 초음파 센서로 출입을 인지하게 된다. 바깥쪽 초음파 센서에 감지된 후 안쪽 초음파 센서에서 감지된다면 사람이 집으로 온 것이고, 그 반대의 경우라면 집을 나간 것이다. 현관에서 출입을 인지할 수 있다는 것은 출입기록을 남길 수 있다는 뜻이고, 외부인의 침입 확인을 카메라를 활용하는 다른 장치들에 비해서 보다 더 쉽고 간단하게 알 수 있어 상대적으로 간단하게 그리고 저렴하게 보안의 기능을 수행할 수 있다. 동시에, 사람이 집에 있을 때만 전기를 공급하고 외출할 때 알아서 잉여 전력을 차단할 수 있는 기능을 제공할 수도 있다. 때문에, 현관에서의 인지는 스마트 홈의 시작이자 끝이 된다고 생각한다.

2) 스피커

사용자가 알림을 인지할 때 스피커에서 나는 소리를 듣는 것이 가장 효과적이고 편리하다고 생각하였다. 또한 음악을 들을 수 있어 사용자의 기분을 좋게, 또는 즐겁게 해줄 수 있다고 생각하였다. 그러하여 스마트홈을 구성할 때 와이파이 기반으로 연결되어 있는 스피커가 필요하다고 판단하였다. MP3 파일을 디코딩할 수 있고 아두이노와 연결할 수 있는 dfPlayer mini 모듈, 소리를 내는 스피커, 그리고 es8266 보드를 이용하여 스피커를 제작하였다. 사용자가 집에 온 것을 현관의 센서를 통해 알게 되었을 때 음악을 재생하고, 집에서 나갈 때도 음악을 재생한다. 만약 다른 알림이 필요할 때도 스피커는 알림 음을 재생할 수 있다. 대쉬보드에서 사용자가 어떠한 분위기의 음악을 듣고 싶은 지 골라 재생할 수 있고, 다음 곡으로 넘기기도 하고 이전 곡을 재생할 수도 있고, 그리고 정지 및 재생도 가능하다.

3) 신발장

보통의 한국인이 집에 들어오자 마자 하는 것은 신발을 벗는 것이다. 하지만 신발 밑바닥은 더럽고 세균에 노출되는 빈도가 제일 높다.[2] 때문에 현관은 세균에 쉽게 더러워지고, 세균에 쉽게 오염된다. 그래서 보통 신발장을 마련하고 신발을 보관한다. 신발장은 밀폐 되어있고 물기가 묻은 신발이 들어간다면 습기가 차기 시작한다. 이런 신발장의 환경은 신발에게는 치명적이 될 수 있다. 때문에 신발장 시스템을 생각하고 계획했다.

스페셜 케어(Special Care)와 노말 케어(Normal Care)로 나눠서 구현했다. 스페셜 케어는 그 날 신은 신발에 대해서, 노말 케어는 다른 보관중인 신발에 대해서 제공한다. 그 날 신은 신발은 당연히 제일 더럽고 세균이 많을 것이기 때문에 FAN과 UV를 통해서 관리할 수 있도록 구성하였다. 다른 보관중인 신발에 대해서는 UV를 제외하고 FAN을 통해서 주기적으로 환기한다. 스페셜 케어는 정해진 시간이 지난 후에는 자동으로 종료가 되고 노말 케어는 주기적으로 FAN이 동작과 정지를 반복한다. 스페셜 케어가 세균 박멸에 필요한 시간 이상으로 작동하는 것은 비효율적이다. 현재 신발을 케어 하는 장치는 UV 뿐이지만 UV가 아닌 다른 것으로도 케어를 한다고 생각해보았을 때, 다시 말해 온도나 다른 장치를 추가한다면 신발에게 오히려 안 좋은 영향을 끼칠 수 있다. 노말 케어 역시 마찬가지로, 주기적으로 몇 일에 한번 정도로 FAN이 돌아간다면 충분하다. 모든 케어는 초음파 센서를 통해 신발이 신발장 안에 있을 때만 작동하게 구성했고, 사용자가 신발을 꺼내기 위해 신발장의 문을 연다면 안전을 위해서 FAN과 UV 모두 작동을 정지할 수 있도록 했다.

[2] 신발바닥 현관에 우글거리는 세균의 정체(코메디 닷컴, 2015.07.01, 미국 휴스턴대 연구 인용)

4) 우산꽂이

우산은 매일 사용하지 않지만, 비가 왔을 때 사용한다. 젖은 우산의 처리가 문제가 되곤 한다. 보통은 펴서 말려야하지만, 아파트와 같은 공동주택에서는 문 앞 복도에 펴놓는 것이 일반적이다. 이는 통행에 큰 불편을 초래하고 복도를 혼란스럽게 만들며 젖게 만든다. 또한 우산을 쉽게 잃어버릴 수도 있다. 하지만 이러한 불편한 점 때문에 현관에 두게 된다면 현관이 더러워지고 습기가 차게 된다. 또한 공간이 협소할 경우 우산을 펴놓지 못해 제대로 마르지 않을 수도 있다. 우리는 이런 불편함을 해소하기 위해서 우산꽂이를 제작했다. PSD 센서로 우산이 감지된다면 FAN이 돌아가서 우산을 말리고 하단에 이 우산에서 떨어지는 물을 받는 물통이 있다. 이 물통에 물이 가득 차게 되면 상단에 LED와 대쉬보드가 표시가 되어 사용자가 교체할 수 있도록 하였다.

5) 칫솔걸이

칫솔걸이에 넣고자 했던 기능은 칫솔의 사용횟수를 파악해주고 사용 후 살균까지 해 주는 기능이었다. 사람들은 칫솔과 면도기를 사용일수가 넘게 사용하지만 교체를 하지 않는다. 교체를 하지 않는 이유는 다양하겠지만 보통 교체를 해 주는 기간을 모르는 사람들이 많고, 다시 새로운 칫솔 혹은 면도기를 꺼내는 것이 귀찮아 교체를 하지 않는다고 생각했다. 일반적으로 사람들은 칫솔이 벌어지거나 면도기가 잘 안 밀릴 때가 되고 나서야 교체를 한다. 여기서 사람들이 간과하고 있는 것은 칫솔이 벌어지고 면도기가 잘 밀리지 않게 되었을 때는 이미 위생적으로 아주 좋지 못한 상태가 되었다는 점이다. 실제 칫솔의 경우 하루 2번 사용한다고 가정했을 시에 3달정도의 사용을 권장하고 있다. 면도기의 경우 매일 사용시에 2주 정도의 사용을 권장하고 있다. 이러한 도구들은 피부 및 치아의 위생에 직접적이기 때문에 세면도구들의 사용일수 혹은 사용횟수가 중요하게 여겨지는 사항이다. 그래서 우리는 사용일수를 파악하고 사용자에게 알림을 주는 제품을 통해 세균의 번식을 막아 피부 질환을 예방하자는 차원에서 제작하게 되었다. 면도기의 경우, 제작이 가능하기는 하지만 우선적으로 칫솔을 만들어 가능성을 확인해 보기로 하였다.

먼저 살균 기능의 경우, 현재 판매되고 있는 많은 칫솔걸이들은 UV램프를 이용하여 살균하고 있었다. 하지만 UV램프의 빛은 인체에 유해하기 때문에 대부분 UV램프의 빛을 사람으로부터 차단하기 위해 뚜껑을 씌운 형태로 제작되었다. 그러나 뚜껑은 사용할 때, 그리고 사용을 하고 난 후 꼭 건드려주어야 하기 때문에 물때가 끼는 현상이 생길 수 있다는 단점이 존재하였다. 그래서 우리는 모션센서를 이용하여 뚜껑을 쓰지 않고 UV램프를 차단하기 위해 사람이 없을 때만 살균을 하도록 설계하였다. 모션센서를 통해 칫솔걸이 주변의 움직임을 파악하고, 만약 움직임이 파악되지 않을 경우 UV램프를 작동시켜 살균을 할 수 있도록 하였다. 결과적으로 뚜껑의 형태를 가지지 않고, 인체에 유해한 UV광선을 차단하는 동시에 살균도 할 수 있는 하드웨어를 제작하였다.

다음으로 칫솔의 사용횟수를 파악하는 기능이다. 사용 횟수를 기록하기 위해서는 칫솔을 인식할 수 있어야 하는데, 그 기능을 위해 칫솔걸이 앞부분 (칫솔의 뒷부분)에 적외선 센서를 고정시켰다. 칫솔이 칫솔걸이에 걸려 있을 때의 거리와 걸려있지 않을 때 적외선 센서가 감지하는 거리가 다르다는 것을 이용하면 칫솔 유무를 알 수 있게 되고, 그것을 통해 사용횟수 파악도 가능하게 된다. 또한 LED를 통해 사용횟수를 사용자가 알 수 있게 하고, 스위치를 통해 칫솔을 교체하였을 때 기기가 새로운 칫솔임을 알 수 있게 해주는 등록기능을 만들었다. 모든 사이즈는 개발자가 현재 사용중인 칫솔의 사이즈를 기반으로 만들었다.

6) 욕실

욕실에서 필요한 기능은 알맞은 습도를 유지시켜주는 기능이다. 습도 조절의 중요성 또한 칫솔걸이와 동일하게 위생과 관련이 있다. 습도가 과한 상태로 방치를 하게 된다면 곰팡이가 생기고 그로인한 세균감염과 악취가 발생할 수 있다. 또한 물기로 인한 미끄러짐 사고도 일어날 수 있다. 그렇기에 습도에 따라 자동으로 환기를 시켜 관리를 해주는 시스템이 더욱이 필요한 것이다. 기존 화장실은 수동으로 환풍기를 작동시키거나 기능이 추가된 곳은 사람이 들어오면 작동하는 방식으로 습도를 조절하였다. '믿똘팀'의 욕실 IOT 시스템은 단순히 FAN을 통해 공기를 순환시키는 역할만 하지만 기술력과 자본이 더해지면 인체에 좋은 이온풍과 겨울철을 대비한 온풍 기능까지도 가능할 것이다. 동작방식은 습도센서를 통해 현재의 습도값을 받고 FAN에 연결하여 습도가 많이 올라가면 FAN이 동작하는 간단한 하드웨어를 만들었다. FAN은 위에 달아주어 뜨거운 공기가 쉽게 빠져나가게 설계했다. 사이즈는 간단한 동작을 보여주는 욕실이므로 칫솔걸이의 동작에 문제가 없을 정도로 미니어처의 형식으로 만들었다.

2.2. Software 구성

1) 현관

아두이노 툴을 이용해 코딩을 작성했다. 초음파 센서가 값을 읽어올 때, 값이 심하게 튀는 경우가 있어서 불안정했기 때문에 이를 방지할 수 있도록 코딩 상으로 필터를 구성했다.

현관에서 감지하는 것은 집으로 들어오는 것과 집에서 나가는 것이다. 그래서 안쪽과 바깥쪽을 정했고 어느 쪽 하나가 감지되고 난 이후에 1초 동안 다른 한 쪽의 감지를 기다리도록 했다.

사람이 들어오고 나갈 때 각각 '1IN'과 '1OUT'를 서버에 전송한다.

2) 스피커

사용한 스피커 모듈은 dfPlayer mini이다. 해당 모듈을 사용하기 위해서 관련 사이트에서 헤더파일과 cpp파일을 사용했다. 내부 구성을 보면 스피커를 구동하기 위한 함수를 잘 구현해 놓았다. 사용을

위해 micro sd카드 내부에 음악파일 이름을 0001, 0002 등으로 변경을 해놓았다. cpp 내부의 함수들, 예를 들어 mp3_play() 또는 mp3_pause()등 필요한 함수를 적절히 활용하여 소프트웨어를 구상하였다.

현관에서 사람이 출입하여 '1IN' 신호 및 '1OUT' 신호를 스피커에서 받게 되면 mp3_play_physical() 함수를 이용하여 특정 음악파일을 재생할 수 있다. 추후 구현할 수 있는 것으로는 mp3_play_file_in_folder (uint8_t folder, uint32_t num) 이라는 micro sd카드 내부의 폴더별로도 재생이 가능한 함수를 이용해 특정 분위기의 노래를 틀고 싶은 경우에도 사용이 가능하게 제작할 수 있다.

3) 신발장

초음파 센서를 통해서 신발을 인식하고 신발이 있을 때만 FAN과 UV가 작동하도록 제작했다. 실제 제품이 된다면 작동 중 문을 열었을 때 여러가지 위험성 때문에 작동이 강제로 중지되어야 하는 점을 나름 중요하게 생각했다. 그래서 이 점을 단순하게 중첩 조건문으로 구현할 수도 있었지만, 좀 더 즉각적인 정지를 위해서 문 쪽에 FAN과 UV가 연결된 스위치를 달았다.

신발이 감지되면 이 스위치에 'HIGH' 값을 준다. 이때 스위치가 눌린다면 FAN과 UV에 전압이 인가되고 작동할 수 있도록 했다. 반대로, 이 스위치가 눌려 있지 않다면 FAN과 UV가 'HIGH' 값일지라도 FAN과 UV에는 전압이 인가되지 않고 작동하지 않게 된다.

그리고 스마트 홈의 주요 기능 중 하나가 효율과 절약이므로 스페셜 케어는 일정시간 동안에만 신발 케어(FAN과 UV)가 작동되도록 했다. 노말 케어칸의 경우, ON/OFF를 반복한다. 물론, 이 모든 것은 신발장에 신발이 있음과 동시에 문이 닫힌(스위치가 눌려 있는) 경우에만 해당한다.

스페셜 케어 칸의 신발이 감지될 때 '3shoesEX', 없을 때 '3shoesNO'를 전송하고 노말케어 칸은 '3shoenEX', '3shoenNO'를 전송한다. 스페셜 케어가 작동할 때 '3start'와 종료될 때 '3finish'를 전송한다. 사용자가 원하지 않을 때 '3fansOFF', '3fannOFF'를 받아 강제 정지할 수 있으며 '3fansON', '3fannON' 명령을 통해 다시 작동이 될 수 있도록 했다. fan(x)ON과 fan(x)OFF는 대쉬보드를 통해 전송받는다.

4) 우산꽂이

우산꽂이는 날씨를 기상청의 RSS에서 가져오도록 했다. ESP8266보드가 직접 하는 것으로, 혹여 서버에 연결되어 있지 않아도 날씨를 표시한다. 대쉬보드를 확인하지 않아도 사용자가 우산꽂이 상단의 LED를 통해서 날씨를 알 수 있도록 했다. 우산이 우산꽂이에 꽂혀 있을 때 FAN을 작동시켜 우산을 건조시키고, 수위센서를 통해 물받이 통의 물을 확인할 수 있도록 했다.

수위센서는 아날로그 값이 아닌 디지털 값으로 받아온다. 이유는 물받이 통을 꺼낼 때 물이 넘칠

수도 있고 수위센서가 젖었을 때 값이 불안했다. 하지만 디지털 값으로 받아올 때 물받이 통을 꺼낼 때 불편하지 않으면서 물을 비워줘야 하는, 80~90% 지점에서 감지했으며 수위센서가 젖었을 때 값이 튀는 것이 보이지 않아서 디지털 값으로 읽어올 수 있도록 했다.

우산이 감지되면 '4umbEX', 없으면 '4umbNO'를 보내며 물받이 통과 관련해서는 '4waterFull', '4waterEmpty'를 전송한다. 대쉬보드에서 전송되어 오는 '4fanON'과 '4fanOFF'를 통해서 사용자의 제어를 받는다.

5) 칫솔걸이

칫솔의 살균을 위해 UV램프를 사용하는데, UV는 인체에 유해한 영향을 줄 수 있다. 즉, 사용자가 있을 때 UV램프가 작동을 하지 않도록 해야 했다. 그래서 다른 센서들보다 UV램프를 작동시키는 모션센서를 더 우선순위로 두었다. 모션센서가 인식이 되고 인식이 되지 않은 경우를 가장 큰 if문으로 하고 그 안에 나머지 칫솔 인식과 관련된 코딩을 넣었다.

여러가지 if문으로 칫솔의 상황을 확인하였다. 먼저 칫솔의 등록(RESET)을 스위치가 눌리면 'RESET 1,2' 값을 받아서 칫솔의 사용횟수인 'count'를 초기화 시키도록 하였다. 그리고 동시에 서버에 'Tooth1DA'를 보내서 대쉬보드에 바로 초기화 된 'count'값을 표기하도록 하였다.

칫솔의 인식은 적외선 값이 인식되어 LOW가 되면 서버에 'Tooth1EX'의 형태로 보내서 '사용자가 칫솔을 보관중이다'라는 상태를 표시한다. 반대로 센서가 'HIGH'로 인식하면 'Tooth1NO'의 문자열을 보내서 사용중의 상태를 표시한다. 각 코드는 칫솔1과 칫솔2가 있으므로 2개의 함수와 변수를 가지고 있다. 그리고 권장 사용일수를 초과한다면 LED를 통해 알림을 주는 if문이 있다.

6) 욕실

습도센서의 사용을 위해 DHT11의 라이브러리를 사용했으며 'dht.setup'을 통해서 설정했다. 잦은 습도 측정이 필요하지 않다고 생각해서 1초에 한번씩 습도값을 받도록 설정했다. 그리고 사용자가 FAN을 원치 않을 때 작동하는 것을 방지하기 위해 서버로부터 '6AUTOON'과 '6AUTOOFF'를 받아서 OFF의 경우 습도가 높아서 FAN이 동작하지 않도록 하였다. ON의 경우는 습도가 일정 수준을 넘어가면 FAN이 동작하도록 하였다.

7) 서버 구성

초기에 서버를 구성할 때 각 클라이언트마다 고유ID가 있을 것으로 판단하고 입력 신호를 분석해 필요한 클라이언트에 전송해주는 방식으로 하여 만약 여러 신호가 왔을 시 신호가 밀리는 것을 막으려고 하였다. 하지만 문제는 클라이언트의 고유 값을 확인할 수가 없다는 것이었다. 그래서

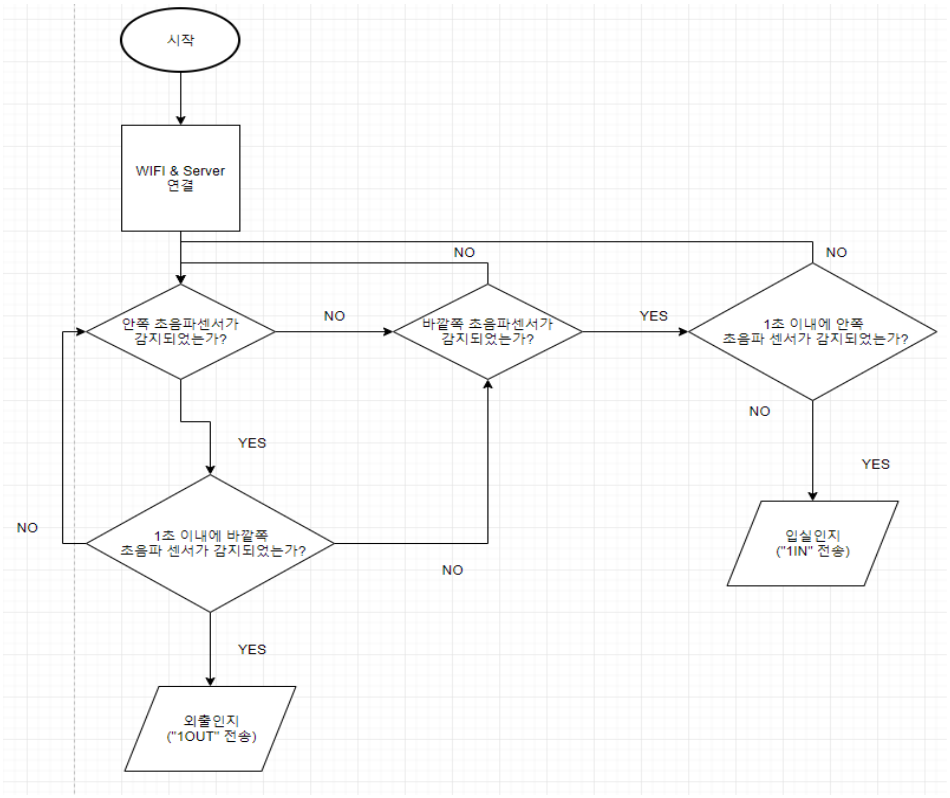
차선책으로 한 클라이언트에서 값을 받으면 그 클라이언트를 제외한 나머지에 다 전송해주는 일명 메시지 서버로 제작하였다.

서버에서 어떤 특수한 일을 할 필요는 없다고 판단했다. 그래서 값을 passing through 해주는 일을 할 수 있도록, 간단하지만 효율적으로 구성하였다.

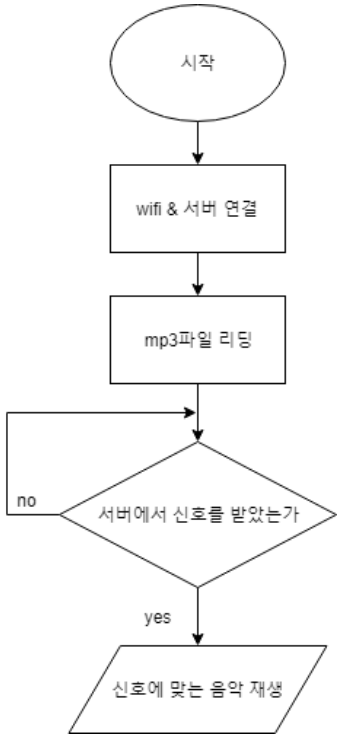
2.3. Software 설계도 (흐름도 및 클래스 다이어그램 등 (개발언어에 따라 선택))

모든 전송부분은 최초 1회만 전송할 수 있도록 하였다.

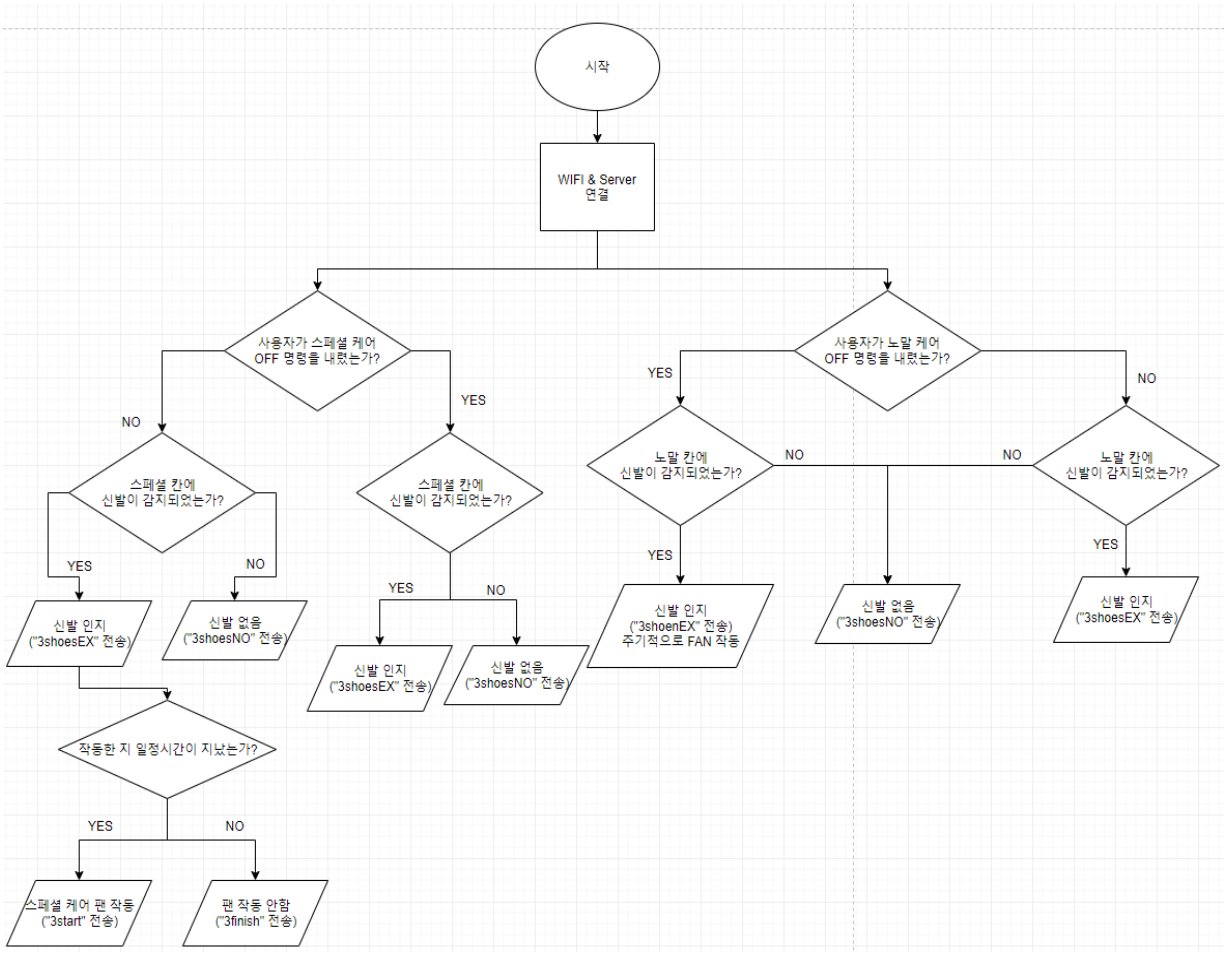
1) 현관



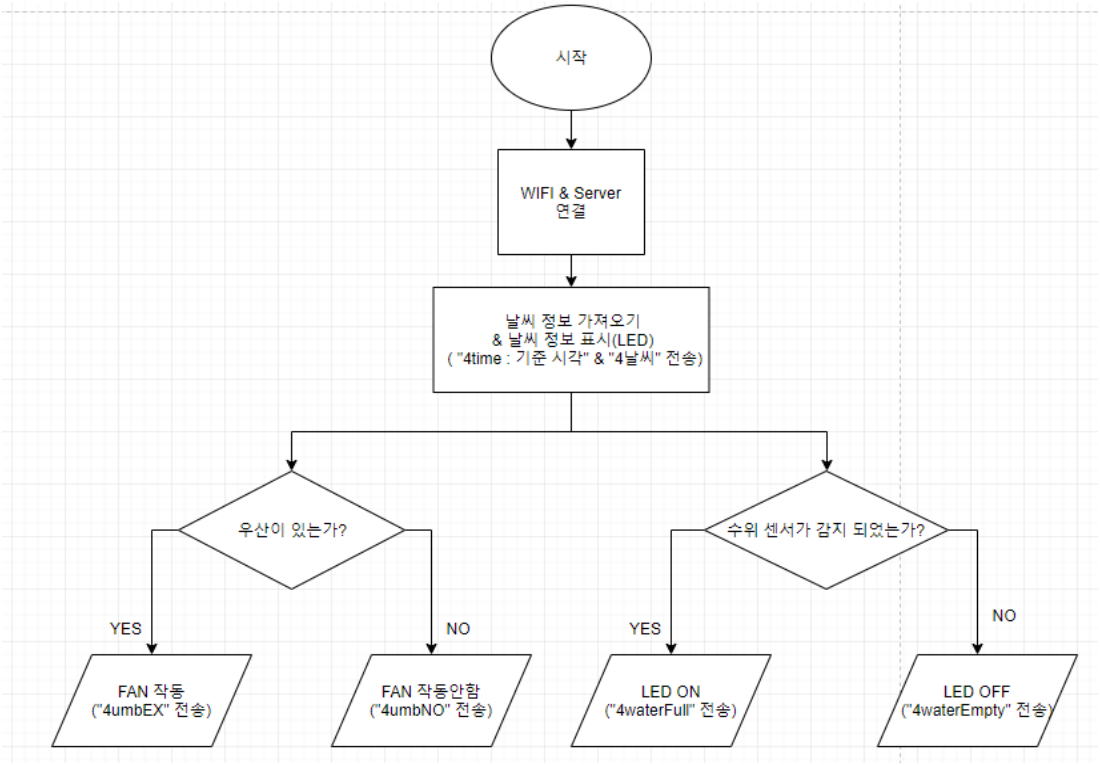
2) 스피커


시작
wifi & 서버 연결
mp3파일 리딩
no
서버에서 신호를 받았는가
yes
신호에 맞는 음악 재생

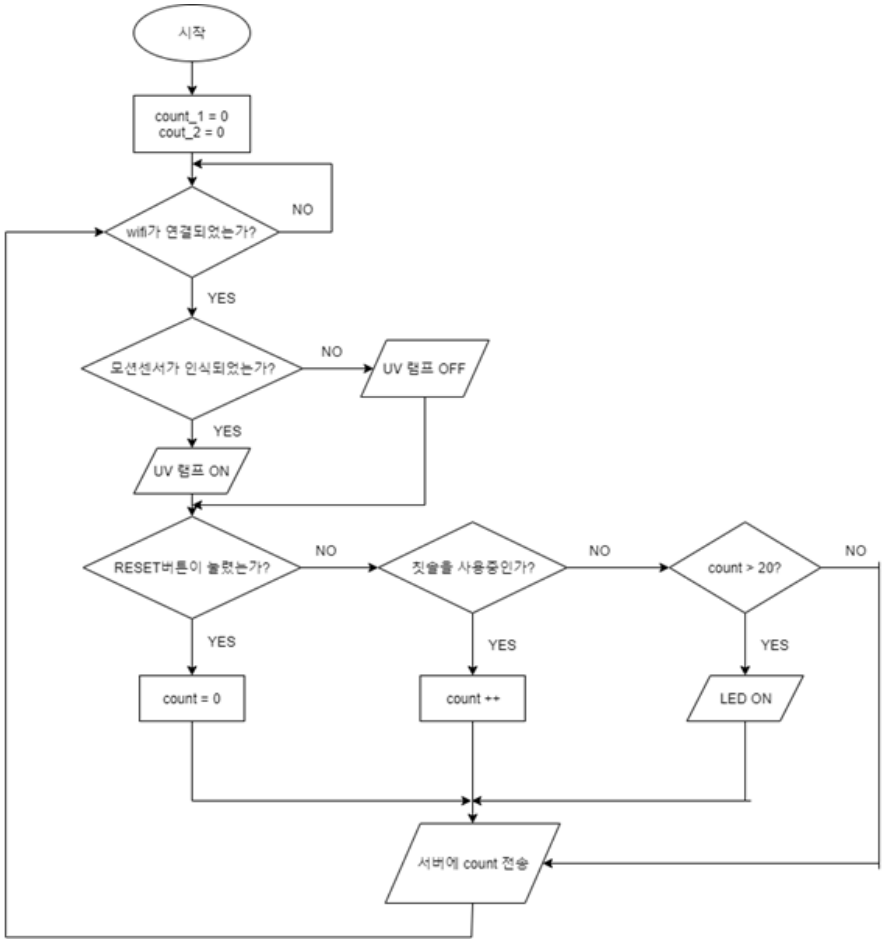
3) 신발장


시작
WIFI & Server 연결
사용자가 스페셜 케어 OFF 명령을 내렸는가?
NO
YES
스페셜 칸에 신발이 감지되었는가?
스페셜 칸에 신발이 감지되었는가?
YES
NO
YES
NO
신발 인지 ("3shoesEX" 전송)
신발 없음 ("3shoesNO" 전송)
신발 인지 ("3shoesEX" 전송)
신발 없음 ("3shoesNO" 전송)
작동한 지 일정시간이 지났는가?
YES
NO
스페셜 케어 팬 작동 ("3start" 전송)
팬 작동 안함 ("3finish" 전송)
사용자가 노말 케어 OFF 명령을 내렸는가?
YES
NO
노말 칸에 신발이 감지되었는가?
노말 칸에 신발이 감지되었는가?
NO
NO
YES
YES
신발 인지 ("3shoenEX" 전송) 주기적으로 FAN 작동
신발 없음 ("3shoesNO" 전송)
신발 인지 ("3shoesEX" 전송)

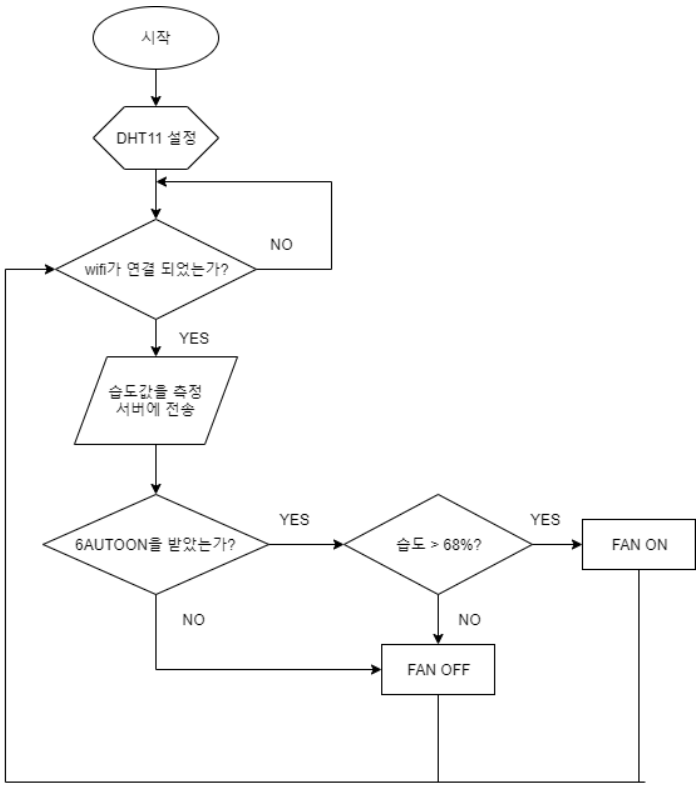
4) 우산꽂이


시작
WIFI & Server 연결
날씨 정보 가져오기
& 날씨 정보 표시(LED)
("4time : 기준 시각" & "4날씨" 전송)
우산이 있는가?
YES
NO
FAN 작동
("4umbEX" 전송)
FAN 작동안함
("4umbNO" 전송)
수위 센서가 감지 되었는가?
YES
NO
LED ON
("4waterFull" 전송)
LED OFF
("4waterEmpty" 전송)

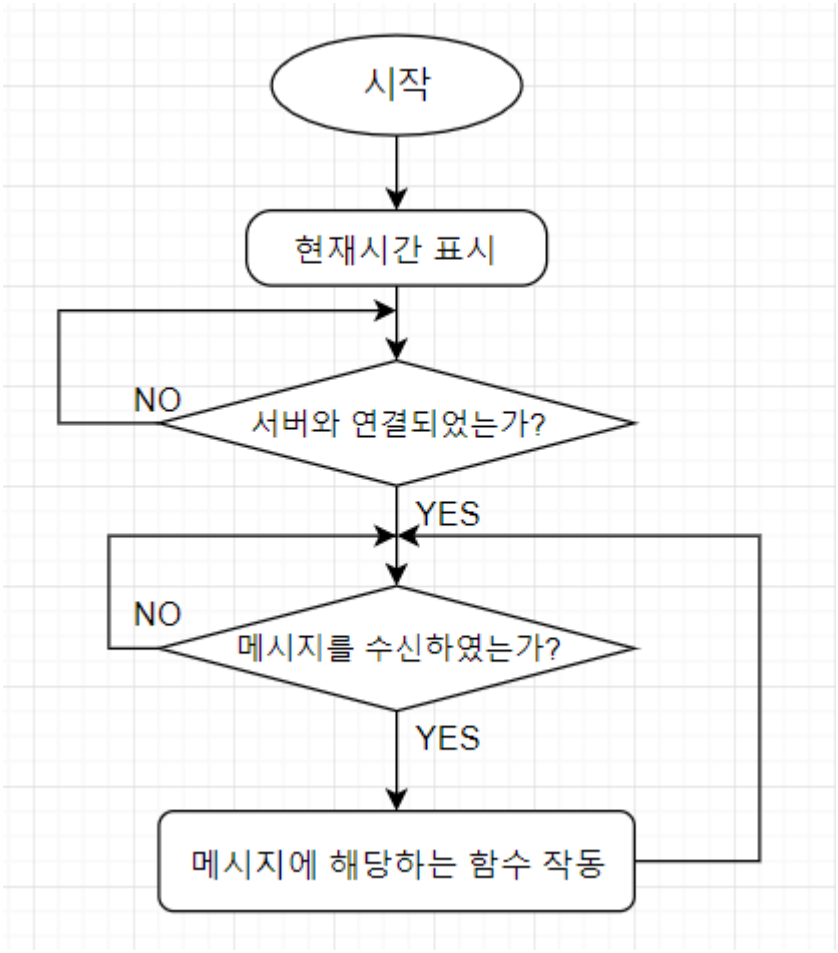
5) 칫솔걸이


시작
count_1 = 0
cout_2 = 0
wifi가 연결되었는가?
NO
YES
모션센서가 인식되었는가?
NO
UV 램프 OFF
YES
UV 램프 ON
RESET버튼이 눌렸는가?
NO
칫솔을 사용중인가?
NO
count > 20?
NO
YES
YES
YES
count = 0
count ++
LED ON
서버에 count 전송

6) 욕실


시작
DHT11 설정
wifi가 연결 되었는가?
NO
YES
습도값을 측정
서버에 전송
6AUTOON을 받았는가?
YES
습도 > 68%?
YES
FAN ON
NO
NO
FAN OFF

7) 대쉬보드


시작
현재시간 표시
NO
서버와 연결되었는가?
YES
NO
메시지를 수신하였는가?
YES
메시지에 해당하는 함수 작동

2.4. Software 기능 (필요 시 알고리즘 설명 포함)

1) 현관

출입을 인지하여 '1IN'과 '1OUT'를 각각 전송한다. 출입을 인지하는 기능은 중요하다. 보안과 효율의 중요한 '키'가 될 것이기 때문이다.

먼저 보안의 측면에서, 우리 작품에서 출입의 로그를 기록할 때 두 명령어를 사용하고 있다. 내가 출입하지 않았을 때의 로그를 확인하는 것도 가능하며 더 발전시킨다면, 스마트폰의 맥주소를 스캔하거나 집의 WIFI가 연결되지 않은 스마트폰이 출입하려 할 경우(보통의 경우 자기 집 WIFI는 자동으로 연결되므로) 사용자에게 현재 현관의 정보를 전송하는 것으로 보안을 강화할 수 있을 거라 생각한다.

효율의 측면에서 보자면, 사용자가 외출했을 때 주요전원을 제외한 나머지 전원이 차단되는 것으로 활용할 수 있을 것으로 생각한다. 날씨 정보와 합쳐져서 사용자가 외출했을 때, 창문을 열어 환기를 하거나 비가 오면 열어놓은 창문을 닫아주는 스마트 창문도 가능하다.

2) 스피커

음악 파일을 재생한다. 소리로 알림을 표현해 주는 것은 대쉬보드를 확인할 수 없을 때 집안의 기능들이 어떤 상태에 있는지 알려주기 때문에 아주 유용하고 효과적이라고 할 수 있다.

현재는 현관의 입출력에 대해서 알림음을 재생하도록 했지만, 더 발전시킨다면 각 IOT 기기들의 동작에도 알림음을 재생해줄 수 있다. 또한, 가장 본질적인 기능인 음악 재생이 가능하다. 와이파이로 스피커가 연결되어 있다는 것은 사용자가 블루투스 및 유선으로 연결하는 것과는 큰 차이가 있는데, 사용자가 스피커 주변에 있지 않아도 제어를 할 수 있어 집에 들어가기 전에 음악을 재생시켜 놓을 수 있고 음악 재생을 재생시켜 놓고 집을 나가도 밖에서 음악을 정지할 수 있다.

3) 신발장

신발이 있음과 없음의 상태를 전송하고 케어 시스템이 작동하고 있는지를 전송한다. 노말 케어는 주기적으로 FAN이 ON/OFF 된다. 그리고 문이 열린다면 스위치가 눌리지 않게 되어 대쉬보드에서는 'ON' 이어도 FAN과 UV는 작동하지 않는다.

신발장은 본연의 역할을 함과 동시에 사용자가 작동여부 명령을 내릴 수 있도록 되어 있다. 스페셜 케어의 일정 시간이라는 조건을 단 이유는 UV를 조사하는 등 추가적인 조치를 취하는 스페셜 케어가 오랜 시간 지속된다면 신발에 좋지 못한 영향이 갈 거라 생각하기 때문이다. 동시에 효율적인 면을 생각해봤을 때, 필요한 만큼만 작동되는 것이 좋다.

4) 우산꽂이

우산이 있음과 없음의 상태와 날씨, 그 날씨의 기준 시각, 그리고 물받이 통의 상태를 전송한다. 우산의 있음이 곧 FAN의 작동을 의미한다. 물론, 사용자가 원한다면 대쉬보드에서 명령을 내려 FAN의 작동을 멈출 수 있다. 날씨는 대쉬보드에서도 쉽게 확인할 수 있게 위함이다. 기준 시각 역시 마찬가지다. 물통은 넘치지 않도록 관리되어야 하기 때문에 대쉬보드에서도 표시하여 사용자가 관리할 수 있도록 했다.

강우량에 따라서 우산이 젖은 정도가 다를 것이고, FAN을 통해서 우산을 말리는 것은 어느 정도 한계가 있다고 생각한다. 그래서 우리의 작품은 일정 시간만 작동할 수 있도록 제작하지 않았으며 다른 추가적인 조치를 취하지는 않았지만 상품화가 된다면 이 부분을 보완해야할 것이라고 생각한다.

5) 칫솔걸이

사용자가 주변에 있고 없음을 파악하고 그 상황에 따라 살균(UV램프)의 작동을 결정한다. 모션센서에 대한 if문을 통과하면 등록, 사용, 경고에 대한 if문이 각각 존재한다. 하지만 그 중에서도 눌렀을 시 바로 반응이 필요한 순서대로 if문의 순서를 배치했다.

1번 스위치를 누르게 되면 RESET_1이 입력이 되고 count_1이 문자열로 변환이 되고 서버로 전송이 되고 대쉬보드에 횟수 0이 표시가 된다. 2번 스위치에 대해서도 동일한 동작을 한다. 이 RESET_1,2는 사용자가 칫솔을 교체하고 횟수를 초기화 할 때 사용하게 된다.

그리고 그 다음 if문은 사용횟수 파악이다. 1번 적외선이 감지가 되어 brush_1이 0이 되면 칫솔이 보관 중이라는 말이 되고 count_1이 증가하지 않는다. Brush_1이 0인 상태가 유지되다가 적외선이 감지가 되지 않아 1인 상태로 바뀌게 되면 칫솔이 사용 중이라고 판단하고 count_1이 증가한다. 그리고 그 count_1값은 '5toothbrush1:DA1'의 문자로 바뀌어서 전송이 되고 대쉬보드에 실시간으로 내가 얼만큼 사용했는지에 대해 표시된다.

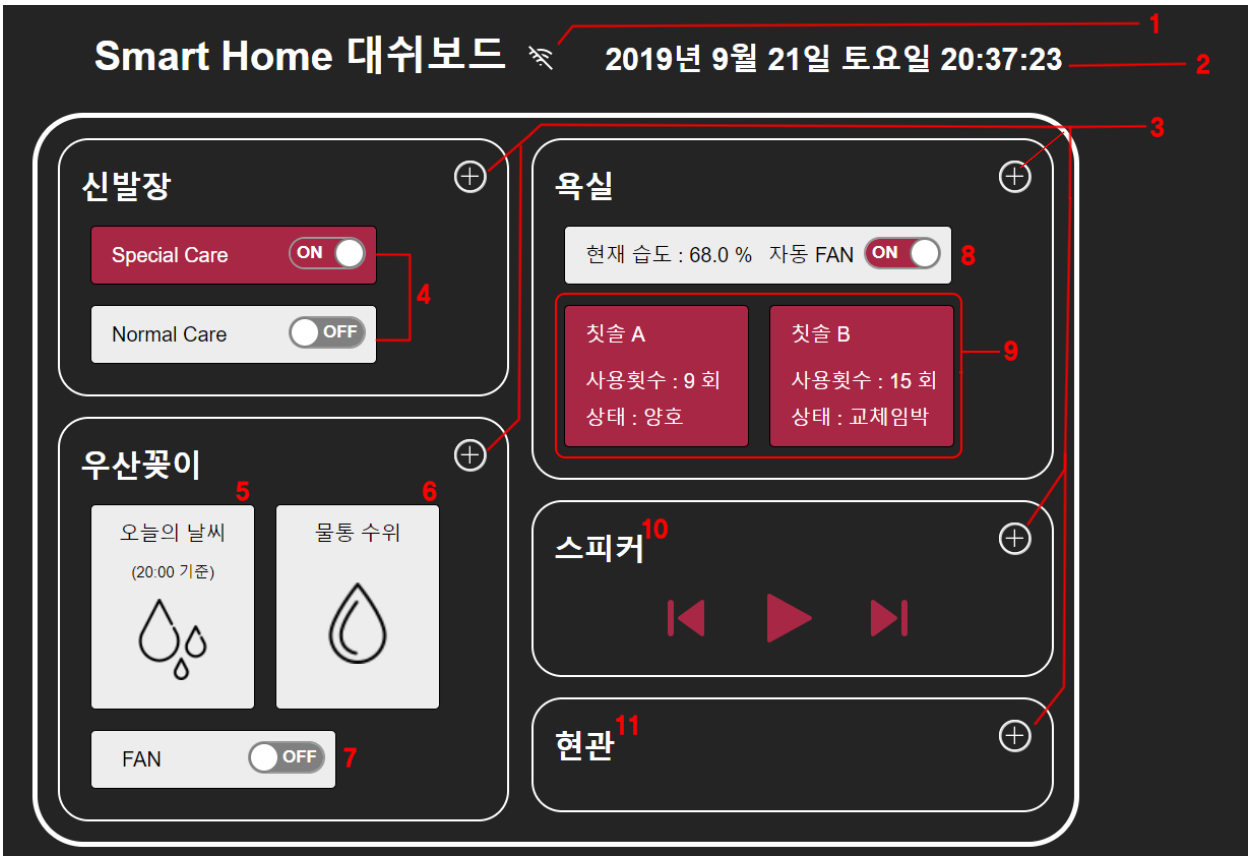
다음 if문은 count_1이 20이 넘어가면 LED가 깜빡이고 대쉬보드에 toast 알림으로 칫솔의 권장사용시기를 초과했다는 알림을 표시한다. 이것도 마찬가지로 '5tooth1:DA1'의 문자열을 전송해서 사용횟수를 바로 보여준다. 이 모든 동작은 2번 칫솔도 동일하게 동작한다. 실제 적정 칫솔 사용횟수는 200회 미만이지만, 작품에서는 빠른 테스트를 위해 20으로 설정했다.

6) 욕실

대쉬보드에서 'FAN 자동모드'를 'ON'상태로 바꾸면 esp8266에 '6FANAUTOON'이라는 문자열을 받고

'fan'이라는 변수에 1로 저장한다. if문에서 'fan'이 1 이면 습도가 68%가 넘어가면 FAN이 동작한다. 반대로 자동모드를 OFF하면 6AUTOOFF라는 문자열을 받고 fan 변수에 0 이 저장되어 다른 if문으로 들어가게 된다. 그 if문은 습도에 관계없이 FAN을 동작하는 핀에 신호를 주지 않는다, 그리고 모드를 변경할 때마다 대쉬보드에 욕실의 FAN이 AUTO모드인지 아닌 지에 대한 알림이 표시된다.

2.5. UI 사용법 (구성 및 설명)



스마트 홈의 핵심은 사용자에게 정보를 제공해 주는 것이다. 우리 팀의 스마트 홈은 현관, 신발장, 우산꽂이, 화장실 및 칫솔걸이로부터 정보를 받아들이고, 이를 대쉬보드에 띄워 사용자에게 전달함과 동시에 이를 사용자가 직접 컨트롤 할 수 있도록 하였다. 메인 대쉬보드를 통해 전체적인 시스템의 작동을 확인할 수 있고, 필요에 따라 사용자가 각각의 시스템의 작동을 정지시킬 수 있다.

우리 팀의 UI (대쉬보드)의 구성은 위와 같고, 각 부분별 기능은 아래와 같다.

1) 연결 상태 표시: 대쉬보드가 서버와 연결되었는 지의 여부를 아이콘으로 표시한다.

2) 현재 시간 표시: OS의 현재 시간을 불러와 표시한다.

3) 더 보기 버튼: 신발장, 우산꽂이, 욕실, 현관은 각각의 고유한 더 보기 페이지를 가지고 있으며, 이 버튼을 통하여 해당 페이지에 접근할 수 있다. 이 기능은 해당 가전과 연관된 다른 정보를 추가적으로 제공하는 기능이 있다.

우리 팀의 시나리오에서는 신발장에서의 더 보기 기능은 신발 소재별 세탁법, 관리법, 신발 추천 등의 연결 링크를 제공하고, 우산꽂이의 더 보기 기능은 기상청과 우산 구매처 링크를 제공한다. 욕실 더 보기에서는 올바른 양치법, 칫솔 관리법, 칫솔 구매처 등의 링크가 제공되며, 현관 더 보기 버튼을 누르면 현관의 출입 기록을 확인할 수 있는 페이지로 이동할 수 있도록 구상했다. 더 나아가 상품화된다면 광고로 자연스럽게 연결되는 등 다양한 방향으로 응용 및 발전될 것으로 기대한다.

4) 신발장 칸 관리: 신발장의 스페셜 케어, 노말 케어 각 칸에 신발이 들어 있는지 여부를 색 변화로 표시하고, FAN의 작동 여부를 ON-OFF 버튼으로 표시한다. ON-OFF 버튼을 클릭하여 직접 FAN을 켜고 끌 수 있으며, 신발장이 칸에 없는 경우엔 버튼이 OFF 상태로 비활성화되어 FAN을 켤 수 없도록 하였다.

5) 우산꽂이 날씨 표시: 우산꽂이에 내장된 esp8266에서 받은 날씨 정보를 아이콘으로 표시하고, 날씨의 기준이 되는 시간도 받아서 표시한다.

6) 우산꽂이 물받이 통 수위 표시: 우산 건조 과정에서 떨어진 물이 모이는 물받이 통이 가득 찼는지 아닌지를 표시하고, 가득 찰 경우 아이콘이 변함과 동시에 toast 알림을 통해 사용자에게 알려준다.

7) 우산꽂이 관리: 우산이 꽂혀 있는지 여부를 색 변화로 표시하고, FAN의 작동 여부를 ON-OFF 버튼으로 표시한다. ON-OFF 버튼을 클릭하여 직접 FAN을 켜고 끌 수 있으며, 우산이 꽂혀 있지 않은 경우엔 신발장과 마찬가지로 버튼이 OFF 상태로 비활성화되어 FAN을 켤 수 없도록 하였다.

8) 욕실 관리: 욕실의 습도를 아두이노로부터 전달받아 표시하고, 습도가 일정 수준 이상일 때 FAN이 자동으로 작동하는 '자동 FAN' 설정을 ON-OFF 버튼으로 켜고 끌 수 있게 하였다.

9) 욕실 칫솔 관리: 칫솔 A와 B가 각각 칫솔걸이에 걸려 있는지 여부를 색 변화로 표시하고, 각 칫솔의 사용 횟수를 아두이노로부터 전달받아 표시한다. 그리고 이를 통해 사용 기간을 산출하여 전문가들이 권장하는 칫솔 사용 기간을 초과하였을 경우 교체할 수 있도록 상태 및 알림을 통해 안내한다.

10) 스피커 관리: 뮤직 플레이어 버튼을 통해 스피커의 음악 재생 및 정지, 다음 곡 및 이전 곡으로 이동 등의 동작이 가능하다.

11) 현관: 아두이노가 사람의 출입을 인식하여 보낸 신호를 받아 toast를 통해 안내하고, 동시에 사람이 들어온 것인지 나간 것인지를 시간과 함께 구글 스프레드시트에 기록한다. 이후 더보기 버튼을 통해 이 출입 기록을 열람할 수 있다.

2.6. 개발환경 (언어, Tool, 사용시스템 등)

1) ESP8266 보드 개발 환경

아두이노(1.8.9) 프로그램을 사용했고 ESP8266 WIFI 관련 헤더파일과 신발장을 주기적으로 ON/OFF

하기 위해서 Timer 헤더파일을 사용했다. 센서의 노이즈를 잡기 위해서 필터를 코드로 작성하였고 날씨 정보를 기상청 RSS를 통해서 파싱하도록 구성했다.

2) 라즈베리파이 개발 환경

UI 사용 공간 (대쉬보드)의 구현은 HTML과 CSS, 자바스크립트 언어를 사용하였다. 개발 도구는 Visual Studio Code를 이용하였고, HTML, JS, CSS파일을 이용하여 UI 공간(대쉬보드)를 구성했다.

우리는 메인 대쉬보드를 나타내는 하나의 HTML(index.html)과 각 공간들의 더 보기 페이지들을 나타낼 여러 개의 html들을 만들었다. 우선 HTML로 UI 공간에 나타낼 항목들을 각각 분할하였다. 이후 표시하고 나타내야 할 문구 및 수치들의 공간을 만들고 클릭 버튼, ON/OFF 스위치와 같은 기능들을 추가하였다. 일정 수치의 정보를 받으면 원하는 그림을 띄울 수 있도록 png파일을 링크 시켰으며, 메인 보드에서 '+'버튼을 클릭하면 추가정보를 제공하는 html 파일을 링크 시켰다. 추가정보 파일에는 스크롤을 하지 않아도 정보를 다 읽을 수 있도록 글자 사이즈를 작게 조정하고 사용자가 필요할 것 같은 정보를 담아 놓았다.

JS파일에는 우리가 사용자에게 전달하고자 하는 정보를 대쉬보드에 메시지 또는 숫자형태로 띄울 수 있도록 만든 함수가 포함되어 있다. 각 공간마다 기능별 함수들을 만들어 특정 정보가 클라이언트에서 왔을 때 대쉬보드에 띄울 수 있도록 하였다.

또한 CSS를 통해 UI공간을 디자인하였다. 모든 html파일의 바탕을 검은색, 글씨는 흰색으로 통일하였다. 전체적인 색감은 WebOS의 색감에서 영감을 얻었다. 각 진 것보다는 부드러워 보일 수 있는 둥그런 디자인으로 디자인했다.

3. 개발 프로그램 설명 (최대한 자세하게 기술)

3.1. 파일 구성

1) 현관

REAL_FRONTDOOR

- 현관에 업로드 된 현관 코딩이 담겨져 있는 파일

2) 스피커

REAL_SPEAKER

- 스피커에 업로드 된 스피커 코딩파일

DFPlayer_Mini_MP3.h

DFPlayer_Mini_MP3.cpp

- MP3 재생에 필요한 header 파일과 cpp파일

3) 신발장

REAL_SHOES

- 신발장에 업로드 된 신발장 코딩파일

4) 우산꽂이

REAL_UMBRELLA

- 우산꽂이에 업로드 된 우산꽂이 코딩 파일

5) 칫솔걸이

bathrooms_webclient

- 칫솔걸이에 업로드 된 칫솔걸이 코딩 파일

6) 욕실

Humi_webclient

- 욕실에 업로드 된 욕실 코딩 파일

7) 대쉬보드

```
kyp@kyp-15Z980-GA7JK:~/vscode/webSocket/wsHome$ tree
.
├── appinfo.json
├── bath.html
├── door.html
├── images
│   ├── cloudy.png
│   ├── connect.png
│   ├── disconnect.png
│   ├── goBack.png
│   ├── loading.png
│   ├── next.png
│   ├── pause.png
│   ├── play.png
│   ├── plus10.png
│   ├── previous.png
│   ├── raindrop-clear.png
│   ├── raindrop-close.png
│   ├── rainy.png
│   ├── smartHome.png
│   ├── snowy.png
│   └── sunny.png
├── index.html
├── shoecare.html
├── shoe.html
├── umbrella.html
├── webOSjs-0.1.0
│   ├── LICENSE-2.0.txt
│   └── webOS.js
├── websocket.css
├── websocket.js
└── websocketMore.css

2 directories, 28 files
```

메인 대쉬보드를 나타내는 파일은 index.html이다. Index파일을 통해 각 공간별 기능을 나타내고

데이터를 표시하는 보드를 나타낸다. 현관, 욕실, 신발장, 우산꽂이 총 4개의 공간의 각각의 html파일을 만들어 '+'버튼 클릭 시 추가정보를 확인할 수 있도록 하였다.

각 함수 별 기능들은 websocket.js 파일을 통해 구현하였다. 메인 대쉬보드는 websocket.css를 통해 디자인하였고, 추가정보를 확인할 수 있는 웹 페이지는 websocketMore.css를 통해 디자인하였다. Images파일에 들어있는 PNG 파일들은 날씨 정보, 우산꽂이의 수위, 뒤로가기 버튼 등 사용자에게 보여줄 데이터와 뒤로가기 버튼, 음악 재생 버튼 등 기능 수행에 사용될 그림 파일들이다.

3.2. 함수별 기능

ESP8266 보드 공통함수

- webSocketEvent(WStype_t type, uint8_t * payload, size_t lenght): 웹소켓에서 어떤 이벤트가 발생했을 때 호출되는 함수. 연결이 됐거나 연결이 끊겼을 때의 상태를 알 수 있게 해주고 서버에서 오는 데이터들을 읽을 수 있다. 배열과 적절한 파싱을 통해서 parameter로 넘어오는 payload를 읽을 수 있고 이 값을 통해서 제어 할 수 있게 하였다.

1) 현관

- inAndout(): 현관에서 출입을 감지하는 함수로 초음파 센서의 감지되는 순서로 출입을 판단한다. 그리고 그 출입을 '1IN'과 '1OUT'으로 서버에 전송한다.

2) 스피커

-mp3_play_physical(): 음악을 재생하는 함수로 괄호 안에 숫자를 입력하면 그 숫자로 시작하는 mp3파일이 재생된다.

3) 신발장

- special(): 스페셜 케어 칸의 작동 함수. 일정시간 동안만 팬과 UV가 돌아갈 수 있도록 한다. 여기서 HIGH값을 출력하지만 해당 기능은 스위치가 눌려야 작동하므로 실제 작동 시간은 설정한 시간보다 짧다. 신발이 감지되거나 케어 시스템이 시작되었을 때, 케어 시스템이 종료가 되고 신발이 감지되지 않았을 때 서버에 해당 정보를 전송한다.

- normal(): 노말 케어 칸의 작동 함수. 주기적으로 FAN이 작동할 수 있도록 하며 역시 스위치가 눌려야 작동한다. 신발이 감지되거나 신발이 감지되지 않았을 때 서버에 해당 정보를 전송한다.

4) 우산꽂이

- umbrella() : 우산을 감지하고 FAN을 돌리는 함수. 우산이 우산꽂이 있을 때와 없을 때의 정보를 서버에 전송한다.

- waterlevel() : 물통의 수위를 감지하고 일정 이상이 되면 우산꽂이 상단의 LED를 키는 함수 역시 해당 정보를 서버에 전송한다.

- GetWeatherInformWIFI() : 기상청의 RSS에서 원하는 데이터를 파싱하는 함수. 파싱해오는 정보를 Serial에 표시하며 서버에 전송한다.

- weatherLED() : 파싱 해온 데이터를 읽고 우산꽂이 상단의 날씨 표현 LED를 켠다.

5) 칫솔걸이

- TOOTHB(): 칫솔의 사용일수, UV램프, 서버로 데이터 전송의 기능을 한다.

6) 욕실

- HU_FAN(): 습도를 측정하여 변수에 저장한다. 저장된 변수를 문자열로 바꾸어 서버에 전송한다. 서버로부터 받은 6FANAUTOON, 6FANAUTOOFF를 변수로 바꾸어 FAN의 동작을 결정한다.

7) 대쉬보드 (websocket.js 파일)

- init() : 대쉬보드 초기(시작) 함수. 웹소켓 통신을 시작시키는 doConnect 함수와 현재시간 표시 기능을 시작시키는 clockInit 함수로 구성된다.

- doConnect() : 서버와 웹소켓 통신을 시작하는 함수. 해당 서버 주소로 WebSocket 객체를 생성하여 서버와 연결되었을 경우, 연결이 끊겼을 경우, 메시지를 받았을 경우, 에러가 발생했을 경우에 각각 해당하는 함수로 연결한다.

- onOpen(evt) : 서버와 연결되었을 경우 동작하는 함수. 연결 상태 아이콘을 '연결됨' 으로 바꾼다.

- onClose(evt) : 서버와 연결이 끊겼을 경우 동작하는 함수. 연결 상태 아이콘을 '연결되지 않음' 으로 바꾼다.

- onMessage(evt) : 서버로부터 메시지를 수신했을 경우 동작하는 함수. Switch문과 if문을 통해 각 메시지에 해당하는 함수로 연결한다.

- onError(evt) : 서버와의 연결에 있어 에러가 발생했을 경우 동작하는 함수. 연결을 중단하고 에러 메시지를 출력한다.

- doSend(message) : 서버로 메시지를 전송하는 함수. Message 변수에 해당하는 내용을 전송한다.

- writeToScreen(message) : 테스트 로그 창으로 메시지를 출력하는 함수. 작품 제작과정에서

테스트하면서 문제점을 찾고 즉각적인 반응과 실험을 위해서 사용되었다.

- clearText() : 테스트 로그 창을 초기화하는 함수. 테스트 용으로 사용되었다.

- doDisconnect() : 서버와의 연결을 끊는 함수. WebSocket 객체를 사용한다.

- changeSpecial() : 신발장 스페셜 케어 칸의 FAN의 상태 (ON/OFF)를 변경하는 함수. FAN이 꺼져 있는 상태일 경우 변경사항이 있을 때 FAN을 켜는 신호를 전송하고 toast로 FAN이 켜짐을 알린다. 반대로 FAN이 켜져 있는 상태일 경우 변경사항이 있을 때 FAN을 끄는 신호를 전송하고 toast로 FAN이 꺼짐을 알린다.

- changeNormal() : 신발장 노말 케어 칸의 FAN의 상태 (ON/OFF)를 변경하는 함수. FAN이 꺼져 있는 상태일 경우 변경이 필요할 때 FAN을 켜는 신호를 전송하고 toast로 FAN이 켜짐을 알린다. 반대로 FAN이 켜져 있는 상태일 때에 변경사항이 생긴다면 FAN을 끄는 신호를 전송하고 toast로 FAN이 꺼짐을 알린다.

- changeBatefan() : 욕실 자동 FAN 설정 여부 (ON/OFF)를 변경하는 함수. 설정되어 있는 경우 클릭 시 설정을 해제하는 신호를 전송하고 toast로 설정이 해제되었음을 알리고, 반대로 설정이 해제되어 있는 경우 클릭 됐을 때 설정하는 신호를 전송하고 toast로 설정되었음을 알린다.

- changeUmb() : 우산꽂이 FAN의 상태 (ON/OFF)를 변경하는 함수. FAN이 꺼져 있는 상태일 때 변경 사항이 있다면 FAN을 켜는 신호를 전송하고 toast로 FAN이 켜짐을 알린다. 반대로 FAN이 켜져 있는 상태일 경우 변경사항이 있다면 FAN을 끄는 신호를 전송하고 toast로 FAN이 꺼짐을 알린다.

- specialShoeIn() : 신발장 스페셜 케어 칸에 신발이 들어왔을 경우 작동하는 함수. 대쉬보드의 스페셜 케어 칸 색상을 바꾸고 FAN 상태 또한 켜짐으로 전환한다. 그리고 toast로 스페셜 케어 시작을 안내한다.

- specialShoeOut() : 신발장 스페셜 케어 칸에서 신발이 나갔을 경우 작동하는 함수. 대쉬보드의 스페셜 케어 칸 색상을 바꾸고 FAN이 켜져 있을 경우 꺼짐으로 전환한다. 추가로 toast로 스페셜 케어가 종료되었음을 안내한다.

- normalShoeIn() : 신발장 노말 케어 칸에 신발이 들어왔을 경우 작동하는 함수. 대쉬보드의 노말 케어 칸 색상을 바꾸고 FAN 상태 또한 켜짐으로 전환한다. 그리고 toast로 노말 케어 시작을 안내한다.

- normalShoeOut() : 신발장 노말 케어 칸에서 신발이 나갔을 경우 작동하는 함수. 대쉬보드의 노말 케어 칸 색상을 바꾸고 FAN이 켜져 있을 경우 꺼짐으로 전환한다. 그리고 toast로 노말 케어가 종료되었음을 안내한다.

- umbIn() : 우산꽂이에 우산이 있을 때 작동하는 함수. 대쉬보드의 우산꽂이 FAN 칸 색상을 바꾸고 FAN 상태 또한 켜짐으로 전환한다. 그리고 toast로 FAN 작동 시작을 안내한다.

- umbOut() : 우산꽂이에서 우산이 빠졌을 경우 작동하는 함수. 대쉬보드의 우산꽂이 FAN 칸

색상을 바꾸고 FAN이 켜져 있을 경우 꺼짐으로 전환한다. 그리고 toast로 FAN 작동 정지를 안내한다.

- waterFull() : 우산꽂이 물받이 통이 가득 찰 경우 작동하는 함수. 대쉬보드의 우산꽂이 물받이 통 칸 아이콘을 가득 찬 아이콘으로 바꾸고 toast로 물통이 가득 찼음을 안내한다.

- tAin() : 욕실 칫솔걸이의 첫 번째 칸에 칫솔이 들어왔을 경우 작동하는 함수. 대쉬보드의 욕실 칫솔 A 칸 색상을 바꾼다.

- tAout() : 욕실 칫솔걸이의 첫 번째 칸에서 칫솔이 나갔을 경우 작동하는 함수. 대쉬보드의 욕실 칫솔 A 칸 색상을 바꾼다.

- tBin() : 욕실 칫솔걸이의 두 번째 칸에 칫솔이 들어왔을 경우 작동하는 함수. 대쉬보드의 욕실 칫솔 B 칸 색상을 바꾼다.

- tBout() : 욕실 칫솔걸이의 두 번째 칸에서 칫솔이 나갔을 경우 작동하는 함수. 대쉬보드의 욕실 칫솔 B 칸 색상을 바꾼다.

- doorIn() : 현관에서 사람이 들어오는 것을 감지했을 경우 신호를 받아 작동하는 함수. 구글 스프레드시트에 'IN' 신호를 현재 시간과 함께 기록한다.

- doorOut() : 현관에서 사람이 들어오는 것을 감지했을 경우 신호를 받아 작동하는 함수. 구글 스프레드시트에 'IN' 신호를 현재 시간과 함께 기록한다.

- humidChange(hum) : 욕실의 습도 정보를 받았을 경우 작동하는 함수. 받은 정보를 대쉬보드 욕실 습도 항목에 표시하고 습도가 높을 경우 toast를 통해 사용자에게 알린다.

- tAchange(num) : 욕실에서 칫솔 A의 사용횟수 정보를 받았을 경우 작동하는 함수. 받은 정보를 대쉬보드 욕실 칫솔 A 사용횟수 항목에 표시하고 횟수에 따른 상태 또한 같이 표시한다. 일단 시연을 위해 10회 미만 양호, 20회 미만 교체임박, 20회 이상 교체요망으로 설정하였다.

- tBchange(num) : 욕실에서 칫솔 B의 사용횟수 정보를 받았을 경우 작동하는 함수. 받은 정보를 대쉬보드 욕실 칫솔 A 사용횟수 항목에 표시하고 횟수에 따른 상태 또한 같이 표시한다. 이 또한 10회 미만 양호, 20회 미만 교체임박, 20회 이상 교체요망으로 설정하였다.

- uTimeChange(num) : 우산꽂이의 날씨 업데이트 시간 정보를 받았을 경우 작동하는 함수. 받은 정보를 대쉬보드 우산꽂이이 오늘의 날씨 칸에 표시한다.

- playChange() : 스피커의 정지/재생 버튼이 클릭되었을 경우 작동하는 함수. 스피커가 정지 상태일 때 클릭되었을 경우 재생 버튼을 정지 버튼으로 바꾸고 스피커 재생 신호를 전송한다. 반대로 재생 상태일 때 클릭되었을 경우 정지 버튼을 재생 버튼으로 바꾸고 정지 신호를 전송한다. PlayCheck 변수를 0과 1로 바꿔가면서 정지 상태인지 재생 상태인지 파악한다.

- toast() : 대쉬보드에 알림을 표시하는 함수. 알림은 화면 오른쪽 상단에 표시가 되며 WebOS 내부 Luna service API를 활용하였다. 함수 내부에 알림을 할 내용을 작성하면 그 내용대로 알림이 표시가

되고 일정 시간이 지나면 사라진다.

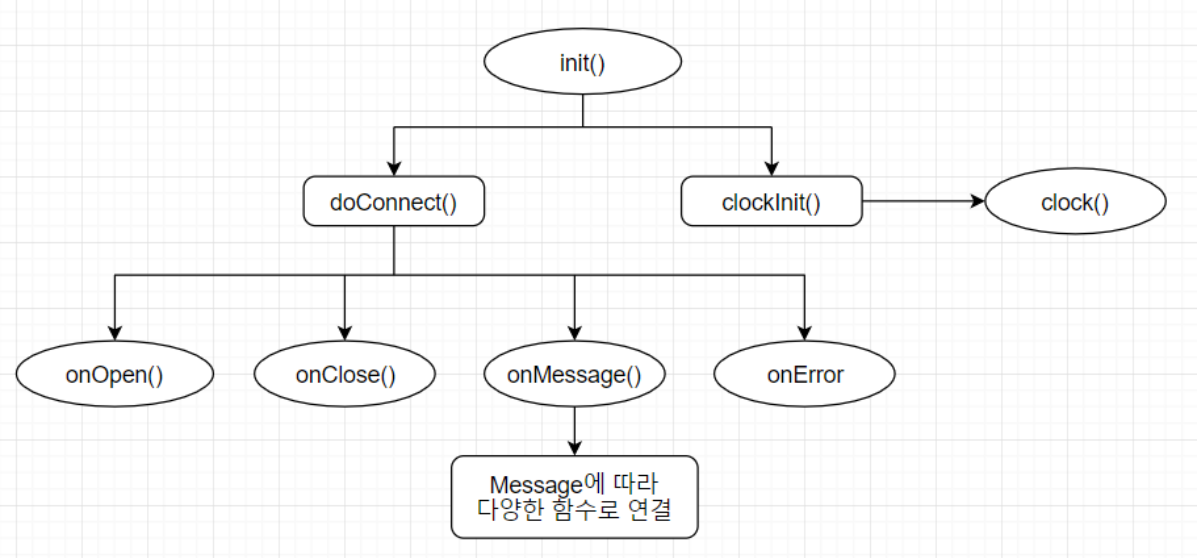
- clock() : 대쉬보드에 현재시간을 표시하는 함수. Date 객체로부터 연, 월, 일, 요일, 시간, 분, 초 정보를 받아서 출력한다. 요일 정보는 0~6의 숫자로 제공되므로 week 배열을 이용하여 요일을 표시한다. 표시 공간을 일정하게 유지하기 위해 시간, 분, 초가 10 미만일 경우엔 앞에 0을 붙여 두 자리 수가 계속 유지될 수 있도록 하였다.

- clockInit() : clock 함수를 1초 단위로 계속 실행하여 현재시간을 1초마다 업데이트한다.

3.3. 주요 함수의 흐름도

우산꽂이 – GetWeatherWIFI() -> weatherLED() : 파싱해온 날씨 정보를 통해서 weatherLED()가 작동할 수 있게 된다.

대쉬보드



3.4. 기술적 차별성

1) 신발장의 설계 - UV램프와 FAN을 작동시킨다는 점, 또한 실제로 상품화가 되었을 때 다른 신발 케어 기능이 추가된다면 사용자가 문을 열었을 때 사용자에게 유해한 영향을 줄 수 있다는 판단을 했다. 그래서 이 것을 소프트웨어적(코드)으로 구현하려고 했으나 전송이 되는 과정이나 감지할 때 딜레이 발생하거나 제대로 작동되지 문제가 발생했다. 그래서 이 것을 해결할 방법으로 하드웨어로 즉각적으로 해결할 수 있도록 했다. 문에 스위치를 달고 이 스위치가 눌려 있을 때만 FAN과 UV에 전류가 흐를 수 있도록 설계했다. 그러므로 케어 시스템은 문이 온전히 닫혀 있을 때만 작동하며 문이 열린다면 작동하지 않는다.

2) 대쉬보드 - 대쉬보드를 통해서 사용자가 IOT기기들을 통제할 수 있다. 신발장, 우산꽂이의 경우

신발과 우산이 있더라도 FAN의 작동이 충분하다고 생각된다면 ON-OFF 버튼을 비활성화시켜 작동시킬 수 없도록 하였고, 같은 정보를 전달하더라도 아이콘 및 글을 최대한 간결하게 하여 한눈에 보기 쉽고 깔끔한 디자인을 구현하였다.

현재 대쉬보드는 WebOS상에서만 존재하지만, 스마트폰 어플리케이션으로 개발하거나 PC 프로그램 등으로 개발할 수 있다면 사용자는 어디에 있던 인터넷만 연결할 수 있다면 자신의 집을 관리할 수 있을 것이다.

3) 하드웨어 인덱싱 – 하드웨어의 인덱싱을 함으로서 우리는 각자 개발해도 하나로 합쳤을 때 문제가 발생하지 않도록 했다. 이 것은 우리가 지향했던 모듈화라는 방향성에 적합하는 것으로 상품화가 되거나 더 발전되었을 때 다른 번호를 붙인다면 혼란이 발생하거나 데이터명을 일일이 확인해야하는 문제는 생기지 않을 것이라 생각한다

4. 개발 중 장애요인과 해결방안

1) 센서의 노이즈

아두이노에서 사용 가능한 값이 저렴한 센서를 사용했는데, 아두이노와 ESP8266 보드의 전압의 차이 (5V와 3.3V), 핀 문제들로 인해서 제대로 센서들의 값이 많이 튀어서 불안정했다. 초음파 센서와 PSD 센서가 특히 그랬는데 이 경우 센서 값을 이전 값과 현재 값의 비율을 적용시켜 연산하여 값을 정하는 방식을 이용해서 센서 안정화를 시켰다. 그 결과 만족할 만한 수준의 정확하고 안정적인 값들을 출력할 수 있게 되었다.

2) 대쉬보드 제작 건

스마트 홈이라면, 내 집의 상황을 파악하는 것이 이루어져야 한다. 그리고 그 다음엔 그 모든 것들을 제어할 수 있어야할 것이다. 그래서 우리는 사용자가 편의를 누릴 수 있는 환경을 생각했고 대쉬보드를 제작하기로 하였다. 그리고 그 구축을 위해서는 WebOS 환경에서 작동할 수 있도록 HTML과 CSS를 활용해야 했다. WebOS 환경도 낯설었지만, html과 CSS 환경에 대해서도 익숙치 않았기 때문에 우리가 원하는 형식의 인터페이스를 대쉬보드를 구축하는 것이 다소 막막했다. 그래서 욕심 부리지 않고 HTML/CSS와 Javascript 기초부터 차근차근 공부하고 계획을 세워가면서 개발을 진행했다.

대쉬보드를 구축하는 것에 있어서 가장 첫번째로 한 작업은 대쉬보드의 구상이었다. 우리의 작품이 많은 가전들을 포함하고 있는 만큼 대쉬보드를 통해서 다양한 가전들을 상태를 확인할 수 있으며 제어할 수 있어야 한다고 생각했다. 그리고 그것을 한눈에 알기 쉽게 표현해야 했다.

가장 어려웠던 점은 이 빈 화면에서 우리가 원하는 모양의 새로운 공간들을 만들고 구성하는

것이었다. 욕실, 신발장, 우산꽂이, 현관, 스피커 총 다섯 가지의 항목을 각각 원하는 사이즈로 구성하기 위해 하나하나를 여러 번의 시도를 통해 경험적으로 조정했다.

또한 웹 페이지 제작 후 이를 웹 앱으로 패키징하여 WebOS로 올리면 WebOS 상에서의 결과는 또 다른 경우가 많아서 이 또한 이것저것 고쳐보고 올려 보고를 반복하면서 경험적으로 수정했다. 이러한 노력들 덕분에 결국 원하는 모양대로 대쉬보드를 구성할 수 있었지만, 분명 더 편한 방법이 있었을 텐데 우리 수준이 부족하여 더 오래 걸리고 번거로운 방법을 택한 것 같아 아쉬움이 있다.

3) WebOS 이용 건

WebOS 내부에는 다양한 API들이 구현되어 있었다. 주어진 API 중 어떤 것을 활용할 수 있는지에 대해 토의했을 때 notification과 정보 저장 API를 사용하자는 의견이 나왔다. 그러나 API 사용 자체에 대해서는 쉽게 알 수 있었지만 이것을 웹으로 구현하는 것에 대해서는 구체적인 자료를 찾기가 힘들었고, 또한 notification 중에 toast는 구현 가능했지만 alert를 구현하는 것은 실패를 하였다.

정보 저장 API는 기본적인 데이터베이스에 대한 지식이 있는 상태에서 사용할 수 있는 것이라고 판단했고 우리 팀에서는 할 수 없었다. 그 외 API들은 WebOS 내부 로그 확인 및 시스템에 대해서 확인을 할 수 있는 것으로 알고 있지만 우리가 구현하는 틀에서 그 정도의 깊이로 접근하는 것이 힘들다고 판단했다. 또한, 그 API들을 이용해 다른 어플들과의 연동을 해야 하는데 우리가 준비한 작품에서는 따로 연동할 수 있는 어플리케이션이 없다는 문제도 있었다.

4) Enact 이용 건

Enact는 WebOS의 UI 개발에 있어서 아주 좋은 툴이라 생각한다. React 기반의 구성과 Import하여 WebOS 내부 파일로 만드는 과정까지 잘 구현이 되어 있었다. 하지만 Enact 만으로는 전체 앱을 짜 보기에는 부족한 것이 많음을 느꼈다.

UI를 구성할 때 사용할 수 있게 마련되어 있는 moonstone decorator의 경우 각종 스위치나 토글 및 앱 활용에 필요한 것들이 잘 구현되어 있지만 실제로 사용을 해 보려고 하니 불편한 것들이 많았었다. 예를 들어 토글(toggle)을 활용하려고 할 때 토글을 직접 조절하는 것 말고도 어떤 값이 들어왔을 때 상태가 변화되는 기능도 필요하다고 생각되었는데 그런 기능이 구현 되어있지 않아서 아쉬웠다. 어떻게 보면 React 와 Enact의 관계가 부모-자식 관계이기 때문이어서 발생한 문제일 수도 있었지만 더 많은 기능이 구현되지 않은 것은 아쉬운 점이라고 생각이 된다.

5) 적외선 센서의 인식

칫솔걸이에서 칫솔의 인식에 사용되는 적외선센서의 인식률이 별로 좋지 못했다. 센서의 인식거리가 정해져 있지만 센서마다 약간의 오차가 있었고 가변저항을 통해 값을 조절하더라도 2~3cm에서 0.1cm 정도 벗어난 위치에 있으면 디지털 값으로는 0과 1을 반복하게 되었다.

이 문제가 실제 칫솔의 뒷부분에 있는 울퉁불퉁한 면에 의해 발생하는 것 같아서 3D 프린터로 실제 사이즈와 동일하게 칫솔을 설계를 하고 뒷부분을 평평하게 센서가 잘 인식하도록 만들었다. 그리하여, 센서의 오차를 줄일 수 있었다.

6) 서버로의 데이터 전송

서버로 칫솔의 사용일수를 전송하여 대쉬보드에서 확인을 하는 과정에서 ESP8266에서 보내는 문자열이 delay없이 계속 전송되어 대쉬보드가 정지하는 현상을 보였다. 아두이노 코딩에서 빠르게 loop문을 돌면서 문자열이 계속 전송되는 것이 문제였다. 그래서 delay를 넣는 것 보다 센서의 값의 변화가 있을 때만 전송하는 걸로 코딩을 변경하였다. 적외선이 인식되는 if문안에 다시 if를 넣고 계속 인식이 되면 임의의 변수 a가 0에서 1이 된다. 그러면 첫 if문의 조건을 만족하지 못하여 다시 들어오지 못 하고 한 번만 값을 전송하게 된다. 임의의 변수 a값은 적외선이 인식 못하는 if문 안으로 들어가게 되면 0으로 초기화되어 적외선이 다시 인식되면 처음의 if문으로 들어갈 수 있게 된다. 이런 방식으로 신호를 한 번만 전송하게 하였다.

이 방법은 다른 가전에서도 적용되어 모든 가전들이 데이터를 보낼 때 한번만 보낼 수 있도록 했다.

7) ESP8266 5V 제어

ESP8266의 경우 5V의 전압이 Vin과 VV 핀에서 나온다. 하지만 여러 센서와 pin들을 연결하면서 5V의 전압이 쉽게 떨어지는 문제가 있었으며 FAN을 돌리려면 5V의 전압에 추가적으로 일정 수준 이상의 전류가 필요했는데, ESP8266에서의 전류는 부족했다. 때문에 이 문제를 해결할 방법이 필요했다.

그 문제를 해결하기 위해서 BJT를 사용했다. 입력의 단자에 센서 값을 주고 5V와 GND 사이에 FAN의 +,-를 연결했다. 그 결과 센서에서 HIGH의 값이 나오면 BJT의 전류를 흐르게 해주어 FAN이 작동했다.

5. 개발결과물의 차별성

1) 실제로 활용될 수 있는 하드웨어의 제작

첫번째는 지금 당장 사용해도 손색이 없는 결과물을 만들었다는 것이다. 비용의 문제로 신발장의 크기가 다소 작지만 우산꽂이와 칫솔걸이는 지금 당장 집에 가져가서 사용해도 될 정도로, 즉각

활용할 수 있는 제품을 제작했다고 생각한다.

'스마트 홈'이란 단어를 접해본 사람은 많지만 지금 적극적으로 활용하고 있는 사람은 별로 없다. 스마트 홈이 대중화되려면 일단 비용적인 부분에서 경쟁력이 있어야할 것이고, 많은 제품이 있어야 한다고 생각한다. 그래서 우리의 작품은 비교적 저렴한 가격으로, 지금 당장 사용할 수 있도록 하는 것을 목표로 제작했다. 지금 여기서 끝이 아니라 더 발전할 수 있다는 것을 생각하면서 제작했다. 예를 들어, 신발장의 UV를 비롯한 케어 시스템이 인체에 위험한 영향을 끼칠 수 있을 거라 생각, 문이 열리면 켜지지 않도록 하드웨어적을 설계했다. 대쉬보드의 '더 보기 페이지', 스피커 역시 이런 관점으로 제작했다.

이런 요소들을 통해 실제로 실생활에 바로 투입되었을 때 제 역할을 할 수 있을 제품들을 만들었다고 생각하며, 이 것을 우리 작품의 차별성이라고 생각한다.

2) 다양한 제품

두번째는 다양한 제품을 만들었다는 것이다. 우리는 사용자가 스마트 홈에 맞춰 살아가는 것이 아니라 사용자가 자기 입맛 대로 스마트 홈을 구축해야 한다고 생각한다. 그래서 우리의 생각으로는 많은 제품을 만들고 사용자가 이런 스마트한 제품들로 직접 자기 집을 꾸미는 것이 앞으로 스마트 홈의 주요 과제가 되지 않을까 생각했다. 이러한 이유로 최대한 다양한 제품을 만들려고 했다.

그리고 대회의 주요 소재가 되는 WebOS를 생각해보았을 때, 많은 제품이 필요하다고 생각했다. '아이폰'이 큰 성공을 거둘 수 있었던 것 중 하나로 앱스토어를 기반으로 한 '앱 생태계' 이고, iOS와 안드로이드를 제외한 다른 스마트폰 운영체제가 거의 없다시피 한 것도 이 생태계에 비롯한다고 생각한다. WebOS가 우리가 낯설게 느끼는 것은 그 전에 사용해보지 못했다는 것인데, 이 WebOS를 통해서 여러가지 제품들을 한번에 제어할 수 있고 WebOS가 제어할 수 있는 그런 제품이 많아진다면 WebOS가 스마트 홈 시장에서 강자로 거듭나지 않을까 생각한다.

6. 단계별 개발계획 및 실제 참여인원 및 업무 분장

1) 전체 계획 및 하드웨어 구상(06/20 ~ 07/15) :

전체적인 계획을 구상했다. 필요 물품의 배송 및 제작기간을 고려해서 하드웨어를 먼저 제작하기로 하였고 하드웨어의 크기와 사용할 센서 등을 토의했다. 토의 결과를 바탕으로 하드웨어 재료와 센서들을 주문했다. 하드웨어 재료가 배송오기 전 까지는 ESP8266 보드를 중심으로 기본적인 회로 및 코드를 구성하였고 실제로 하드웨어를 제작했을 때의 위치와 선의 길이를 계산했다. ESP8266 보드 내에 구현한 아두이노 소프트웨어 소스코드의 경우, 이 단계에서 전체적인 틀이 잡혔다.

(전 인원 참가)

2)하드웨어 제작 및 소프트웨어 구상 단계(07/15 ~ 07/31):

하드웨어 재료 배송이 완료되는 직후 하드웨어를 제작하였다. 기존에 구상했던 회로를 제작하고 재료들을 이용해 하드웨어를 만들고 회로와 하드웨어를 결합하고 납땜하였다. 하드웨어는 이 단계에서 모두 완성되었다. 소프트웨어 구상은 웹서버를 통한 통신과 WebOS 환경에 익숙해지기 위한 학습, HTML과 CSS 및 JavaScript를 공부하기로 했다. 소프트웨어 쪽의 개발환경과 툴은 모두 처음 접하는 것이었기 때문에 자료를 공유하고 서로가 서로에게 멘토가 되어가며 같이 공부했다.

(07/25까지 전원 하드웨어 개발에 참가, 이후 김유현은 WebOS 환경에 익숙해지기 위해서는 무엇을 미리 알아야 하는지 조사했고 통신환경 구축을 위한 여러가지 시도를 했음)

ESP8266 보드 코딩 수정 및 대쉬보드 구상 단계(08/01~08/20):

웹 소켓으로 통신하기로 결정한 이후에는 ESP8266 보드를 전면 수정, 통신이 이루어지도록 했다. 이 때 서버를 구축하는 데에 있어서는 ESP8266 보드, GitHub 등 여러가지 아이디어가 제시되었지만 AWS(Amazon Web Service) 활용으로 크게 안정화되었다. 이 단계에서 대쉬보드의 전체적인 틀을 잡았다. 이 시기에는 여름 휴가철이었고 각자의 사정이 있던 시기였으므로 일정을 넉넉히 잡고 진행했다.

(조성익, 정현우는 코딩 수정을 전담했고 김유현은 서버 구축과 통신 환경에 대해서 작업했다. 김영평, 성기정은 대쉬보드 제작을 진행했다.)

소프트웨어 제작 및 하드웨어 보완(08/21 ~ 09/15):

서버를 구축했으며 대쉬보드를 완성했다. 대쉬보드가 완성되어가면서 혼란이 야기되지 않도록 하드웨어의 번호를 붙였다. ESP8266과 주고받는 데이터를 구분할 수 있도록 계속해서 명령어들을 정리했다. 사용자의 입장이 되어 보면서 가전을 사용할 때 대쉬보드를 통해서 필요하거나 유용한 기능들을 생각했다. 때문에, 하드웨어 보완 작업이 계속해서 이루어졌다. 전체적인 틀이 흔들리지는 않는 디테일을 잡아가는 과정이었다. 이 시기에 Enact와 Luna service API 등 최대한 여러가지를 사용해보면서 최적화 과정을 진행했다.

(김영평, 성기정은 대쉬보드를 제작했다. 김유현은 대쉬보드에 쓰일 수 있는 여러 재료를 찾았고 조성익, 정현우는 하드웨어를 보완했다.)

전체 연동 확인 (09/15 ~ 09/20):

개별이 아닌 모든 하드웨어를 연결하고 이 때 어떤 문제가 발생되는 지를 확인했다. ESP8266 코딩은 물론 대쉬보드 코딩도 계속해서 최적화하고 정제하는 과정이었다. 이 단계에서 여러가지 작업을 동시에 하였을 때 하드웨어의 센서가 제대로 작동하지 않는 문제가 발생하기도 하였다. 기준값을 내리거나 센서를 교체하는 작업을 거쳤고 코딩도 많은 수정을 거쳤다.

(조성익은 신발장, 우산꽂이, 현관을 전담하였고 정현우는 욕실과 칫솔걸이를 전담하였다. 김유현은 스피커와 대쉬보드의 Alert, Toast 기능, 통신을 담당했다. 성기정, 김영평은 대쉬보드를 전담했다.)

완성, 시연 영상 및 보고서 작성 (09/20 ~ 09/28):

이 단계에서는 불안한 부분이나 아쉬운 부분에 대한 디테일을 계속해서 보완해나가는 작업을 했다. 여러 센서에 대해서 필터 작업을 진행했고 호환이 문제가 있는 경우 배제하는 등 수정에 수정을 거듭했다. 만족할 만한 수준이 되었을 때 완성 선언 후 시연 영상을 찍었고 제작했다. 보고서는 영상 찍은 직후에 지금까지의 개발 과정을 뒤돌아보면서 파트를 나누고 대략적인 뼈대를 잡았다.

팀장 조성익은 전체적인 일정을 조정하고 장기적인 목표를 정하고 그 것을 위해서 단기적인 목표를 제시하여 프로젝트가 나아가게 했다. 신발장과 우산꽂이의 아두이노 코드를 작성하였고 현관 아두이노 코드의 초안을 다듬어 통신이 가능하도록 했다. 모든 제품이 웹소켓을 통해서 통신할 수 있도록 했고, 제품마다 번호를 붙여 중복되는 명령어를 쓰더라도 충돌이 일어나지 않도록 했다.

팀원 김영평은 현관 아두이노 코드의 초안을 작성하였고 HTML, CSS, JavaScript를 이용하여 대쉬보드를 웹 사이트의 형태로 제작, 이후 웹 앱의 형태로 패키징하여 WebOS에 업로드하고 잘 작동하는지 확인하는 역할을 전담하였다.

팀원 김유현은 주어진 WebOS 내 Luna service API 사용법, Enact 활용법 탐구, 서버 구축, 현관 입출력 시 정보를 저장하는 방법 찾기 및 스피커 코드작성을 하였다.

팀원 성기정은 사용자 인터페이스를 만들고 꾸미는 역할을 하였다. 대쉬보드의 제작을 담당하였고, 사용자에게 보여줄 정보를 올바르게 보여줄 수 있도록 공간을 만들었고 사용자의 필요에 따른 추가정보를 보여주는 html 파일을 만드는 역할을 하였다.

팀원 정현우는 욕실과 칫솔걸이의 아두이노 코드를 작성하였고 솔리드웍스와 3D프린터를 통한 하드웨어의 설계와 작업 중 하드웨어의 채색 및 수리를 했다.