**무선 종합 경보기**

**최종 보고서**

**▪ 수강과목 : IoT 시스템 설계**

**▪ 담당교수 : 조윤석**

**▪ 학 부 : 전산전자공학부**

**▪ 성 명 :**

**김민수 (21300092)**

**추교찬 (21300794)**

**한찬솔 (21400802)**

**이경민 (21500463)**

**이아름 (21500514)**

**김태연 (21600166)**

**유현아 (21700472)**

**▪ 제출일시 : 2019. 12. 19**

|  |
| --- |
| **목 차**   1. 서론    1. 문제배경    2. Goals and Objectives    3. Constraints 2. 본론    1. 사용된 컴포넌트에 대한 설명    2. Overall System Architecture    3. 기능 구현 및 설명       1. 하드웨어       2. 소프트웨어          1. Level1          2. Level2          3. Level3    4. 시뮬레이션 결과    5. 프로젝트 중 발생한 문제점과 문제 해결 과정 3. 결론 4. 부록    1. Individual Project Assignment |

1. 서론
   1. 문제배경

우리는 매체를 통해 날마다 많은 사건 및 사고 소식들을 접하며 때로는 직접 사고를 경험하기도 한다. 북한 역시 크게 다르지 않은데, 최근 북한 지역의 모든 기관, 기업소 및 가정에서 화재와 가스, 도난 사고가 빈번하게 발생하고 있다. 사고가 난 곳이 어디든지 간에 사고에 대한 대응이 굉장히 중요하지만 그보다 더 중요한 것이 바로 최대한 빠르게 그 사고에 대해서 아는 것이다.

먼저, 북한의 화재 사고에도 많은 원인이 있지만 최근에는 누전으로 인한 화재가 자주 발생하고 있다. 이는 가정뿐만 아니라 큰 기업과 기업소에서도 직면하고 있는 문제이다. 실제로 평양 시내는 하루에 3~4시간정도만 전기가 공급되는 심각한 상황에 있다. 이러한 이유로 오랜 시간 전기 공급이 가능한 곳에서 전기를 끌어다 사용하는 ‘전기 도둑’이 횡행하고 있으며, 안전하지 않은 방법으로 전기를 끌어다 사용하기 때문에 화재의 직접적인 원인이 되기도 한다.

두 번째, 북한지역의 도난사고의 경우, 농민시장이 장마당의 형태로 바뀌면서 제일 빈번하게 도난 사고가 발생하는 품목은 곡류에서 자전거와 컬러TV등의 물품으로 바뀌었다. 또한 장마당이 발전하면서 장사를 잘하는 사람과 못하는 사람이 갈리게 되고, 빈부격차가 생겼으며, 대다수 북한 주민들은 은행에 적금해도 돈을 제때에 주지 않기 때문에 돈을 집에 직접 보관하고 있어 최근에도 북한은 도난과 관련된 각종 범죄가 끊이지 않고 있다. 북한에서는 지금도 자전거를 밖에 놔두는 집이 없으며, 자전거를 어깨에 메고 20층 아파트를 올라가는 것이 일상이 된 지 오래다.

세 번째, 북한의 가스사고 역시 빈번하게 발생한다. 북한에서 가장 생활수준이 높은 평양에서조차 중심지를 제외한 평양 외곽의 상당수 주민들이 연탄을 주 연료로 사용하는 전근대적인 주거환경에서 생활하기 때문에 일산화탄소에 의한 사망사고가 끊이지 않는다. 평양의 중심지는 LPG를 사용하여 난방과 취사를 해결하므로 연탄가스로 인한 사고발생률은 낮지만, LPG를 사용하기 때문에 화재 및 가스사고에 필수적으로 대비를 해야 한다. 하지만 화재, 가스, 도난 사고를 검출하는 시스템이 잘 구축되어 있지 않아 사고를 감지하는데 제한이 있으며, 사고를 감지하더라도 사용자에게 사고상황을 전달할 방법이 부족하다.

* 1. Goals and Objectives

우리는 사람들이 사고 상황을 최대한 빠르게 전달받을 수 있는 굉장히 간단하지만 영향력 있는 시스템을 개발하고자 한다. 한 공간에만 국한되는 것이 아니라 여러 공간에서 발생하는 사고를 동시에 감지할 수 있는 시스템을 개발하는 것이 가장 큰 목표이며, 이를 구현하기 위해 서버와 여러 클라이언트로 시스템을 구성한다. 다양한 클라이언트를 통해 사고가 감지되면 사고 발생 지점의 위치 정보는 물론, 발생한 사고가 무엇인지에 대한 정보까지도 사용자가 알 수 있도록 한다. 사고를 감지하는 모듈인 클라이언트에는 사고의 발생요소를 감지하는 역할을 하는 각종 센서를 탑재하여 화재, 가스, 도난 사고와 문 열림 발생을 감지한다. 사고가 감지되면 여러 액츄에이터(Buzzer, LED, LCD)를 사용하여 사람들에게 현재 상황이 전달되도록 하며 통상적으로 사람이 많이 모이는 장소에 서버를 설치하여 정보 전달의 사각지대가 없도록 한다. 나아가 Wifi 기능을 활용하여 특정 어플리케이션을 알림 기능을 통해 해당 장소에 사람이 부재할 경우도 대비한다.

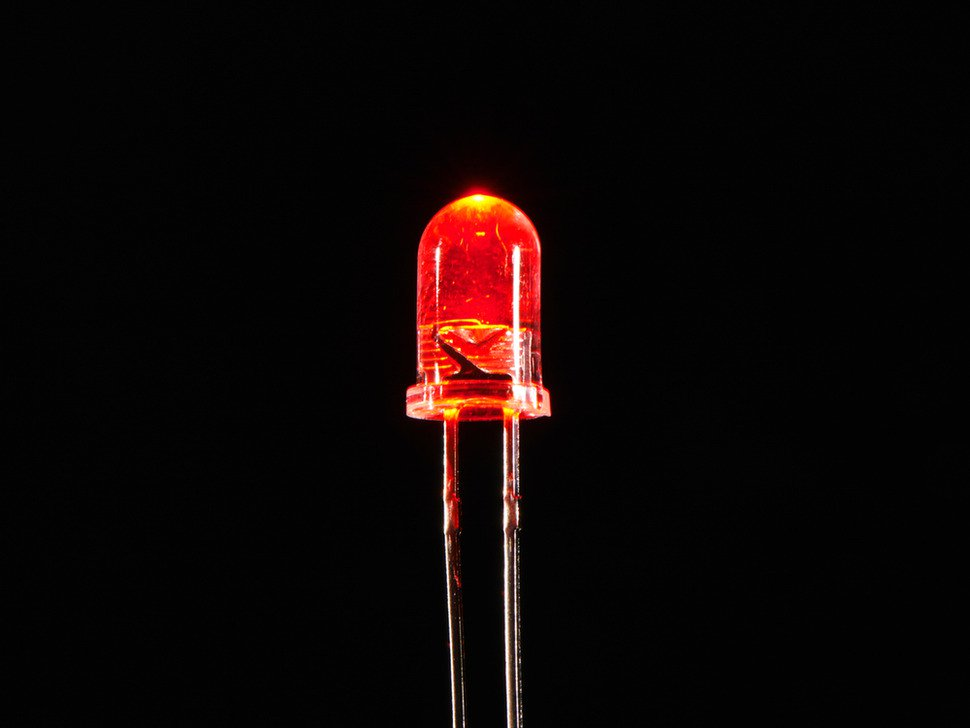
사고가 발생하지 않은 일반적인 경우에 온도와 습도를 출력하며, 사고가 발생할 경우에는 먼저, 발생 지역의 사고 감지 모듈을 통해 발생 지역에 있는 사람들에게 사용자에게 사고상황이 전달된다. 이후, 해당 사고의 발생 위치 정보와 사고상황이 서버로 전달되어 더 많은 사람들이 사고에 대해 인지할 수 있도록 한다.

* 1. Constraints

제시된 제한조건은 DC 12V, 연결 가능한 말단 개수 20~100개이며, 등록가능한 전화번호는 10개, 마디들 사이의 신호전송거리는 100m이다.

현재 진행된 프로젝트에서는 MCU로 NodeMCU를 사용하므로 5V를 필요로 하기 때문에 DC 5V를 공급하며, IFTTT를 사용하여 사람들에게 notification을 보내기 때문에 현재 등록 가능한 전화번호에 제한이 없다. 다만 Wifi로 여러 기능을 동시에 구현해 놓아서 엄청난 패킷의 전달이 필요할 경우 제한이 발생한다.

1. 본론
   1. 사용된 컴포넌트에 대한 설명
      1. LED



<그림 2> Red LED

출처: <https://www.adafruit.com/product/297>

LED(Light Emitting Diode)는 발광 다이오드로 순방향으로 전압을 가했을 때 발광하는 반도체 소자이다. 여러가지 색의 LED가 있는데 현재 시스템에서는 RED LED를 사용한다. LED 사용 목적은 경보음 수신에 제한이 있는 사용자의 경우에는 경보음만으로 사고를 감지하는 것이 힘들기 때문에 LED를 켜서 사고가 감지되었다는 것을 전달하기 위해 사용한다. 서버인 중앙 모듈부터 여러 클라이언트까지 각 모듈이 전부 LED를 포함하고 있으며, 사고가 감지되면 ‘Active-HIGH’로 동작하여 LED가 켜지도록 회로를 구성하였다.

* + 1. Buzzer



<그림 3> 피에조 부저

출처: <https://maltarotors.com/product/5v-piezo-buzzer/>

Buzzer는 소리 신호 알림 장치이며, 이 시스템에서는 압전 방식을 사용하는 피에조 부저(Piezo buzzer)를 사용하였다. 사고가 감지되면 서버와 클라이언트에서 동시에 소리 신호가 출력되며, 이를 통해 사용자가 사고 상황을 인지할 수 있다. 부저는 LED와 같이 두 개의 핀으로 구성되어 있는데 하나는 GND, 나머지 하나는 Digital out pin에 연결하여 사고 상황에 따라 Digital out pin의 출력을 설정하여 소리 신호를 만든다.

* + 1. Nokia 5110 LCD

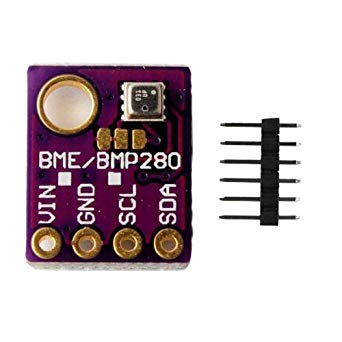


<그림 4> Nokia 5110 LCD (I2C)

출처: <https://www.indiamart.com/proddetail/nokia-5110-lcd-display-module-20384529848.html>

LCD(Liquid Crystal Display)는 액정 디스플레이 또는 액정 표시 장치로 디스플레이 장치의 하나이며, 평판 디스플레이(FPD)의 한 종류이다. Nokia5110 84X48 LCD는 84X48의 픽셀 수를 가지고 있으며, 저전력의 CMOS LCD 제어용 드라이버인 PCD8544 마이크로 컨트롤러를 사용한 LCD이다. 문자 입력뿐만 아니라 한글이나 도형 표현이 가능하다. 이 시스템에서는 SPI 통신 방식을 이용하고 있는 LCD를 사용하여 다양한 정보를 액정에 출력한다.

* + 1. BME280

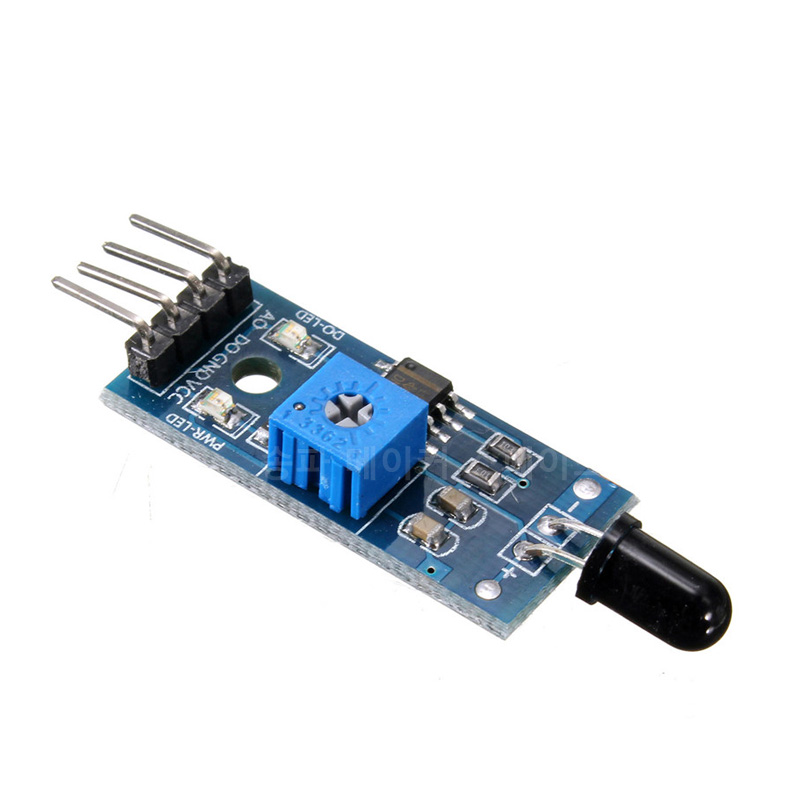


<그림 5> BME280

출처: <https://www.amazon.com/KeeYees-Temperature-Humidity-Atmospheric-Barometric/dp/B07KYJTNR6>

시중에는 다양한 온습도 센서들이 있는데, 이 시스템에서는 BME280을 사용한다. 습도 측정 범위는 ‘0 ~100% RH (±2% RH)’이고, 온도 측정 범위는 ‘-40~80 (±0.5)’이다. 저전력이라는 특징을 가지고 있으며, 사용방법이 굉장히 간단하다는 특징을 가지고 있다. 핀은 VCC, GND, SDA, SCL 4개로 구성되어 있으며 I2C연결방식을 사용하여 쉽게 데이터를 얻을 수 있다는 장점이 있다.

* + 1. 불꽃 감지 센서



<그림 6> 불꽃감지센서

출처: <http://songpamakers.com/mall/product/%EC%95%84%EB%91%90%EC%9D%B4%EB%85%B8-%EB%B6%88%EA%BD%83-%EA%B0%90%EC%A7%80-%EC%84%BC%EC%84%9C/>

불꽃 감지 센서란 말 그대로 불을 감지하는 센서를 말한다. 정확하게 말하면 적외선 LED를 통해 화재 시 연소 반응에 의해 불꽃에서 파생되는 열 복사인 적외선 파장을 감지한다. 근접한 거리의 불꽃만 감지된다는 특징이 있으며, 최대 감지거리는 모듈에 따라 다르지만 현재 시스템을 구성하고 있는 모듈의 경우 17~18츠 정도로 확인된다. VCC, GND, DATA 이렇게 3가지 핀을 가지고 있는데 불꽃의 감지유무에 따라 DATA 핀에서 HIGH(불꽃이 감지되지 않음) 혹은 LOW(불꽃이 감지)의 데이터가 출력된다.

* + 1. 가스 감지 센서



<그림 7> 가스 감지 센서 (MQ2)

출처: <https://www.indiamart.com/proddetail/mq2-gas-sensor-module-19274547712.html>

시중에 다양한 종류의 가스 감지 센서가 있는데 이 시스템에서는 ‘MQ-2’ 모듈을 사용한다. 이는 내부에 포함된 히터와 센서로 LPG, 부탄, 메탄, 알코올 등을 검출해낼 수 있는 모듈이다. 내부에 포함된 히터가 적정온도로 올라가서 안정적인 결과를 얻기 위해서는 대기시간이 조금 필요한다. VCC, GND, Analog out 이렇게 3가지 핀을 가지고 있는데, 가스 농도에 따라 Analog 핀으로 통해 아날로그 데이터가 전달된다. 여러 근거를 통해서 일정 농도 기준을 세우고 그 기준에 따라 가스 사고 유무를 판단한다.

* + 1. 문 열림 마그네틱 센서

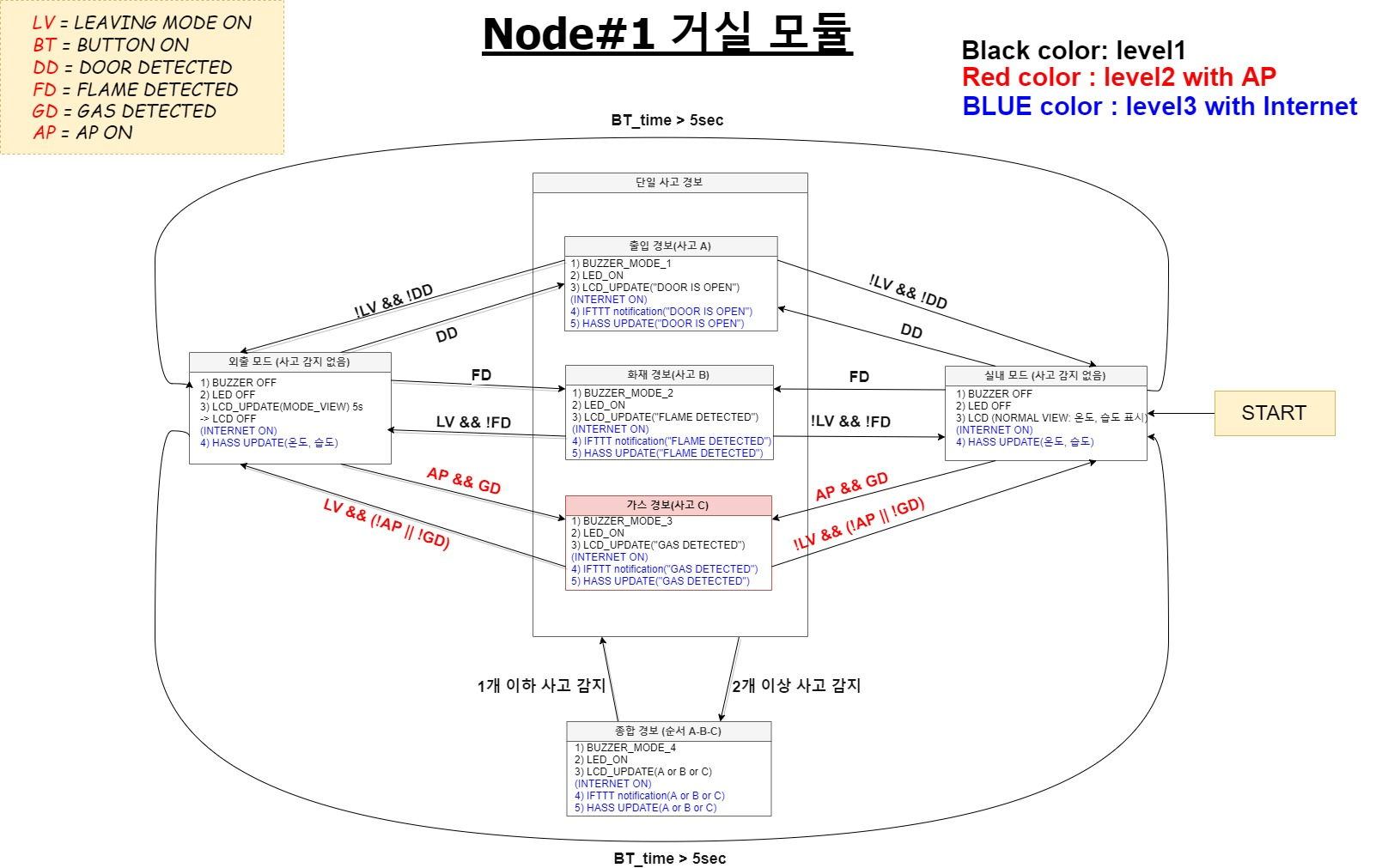


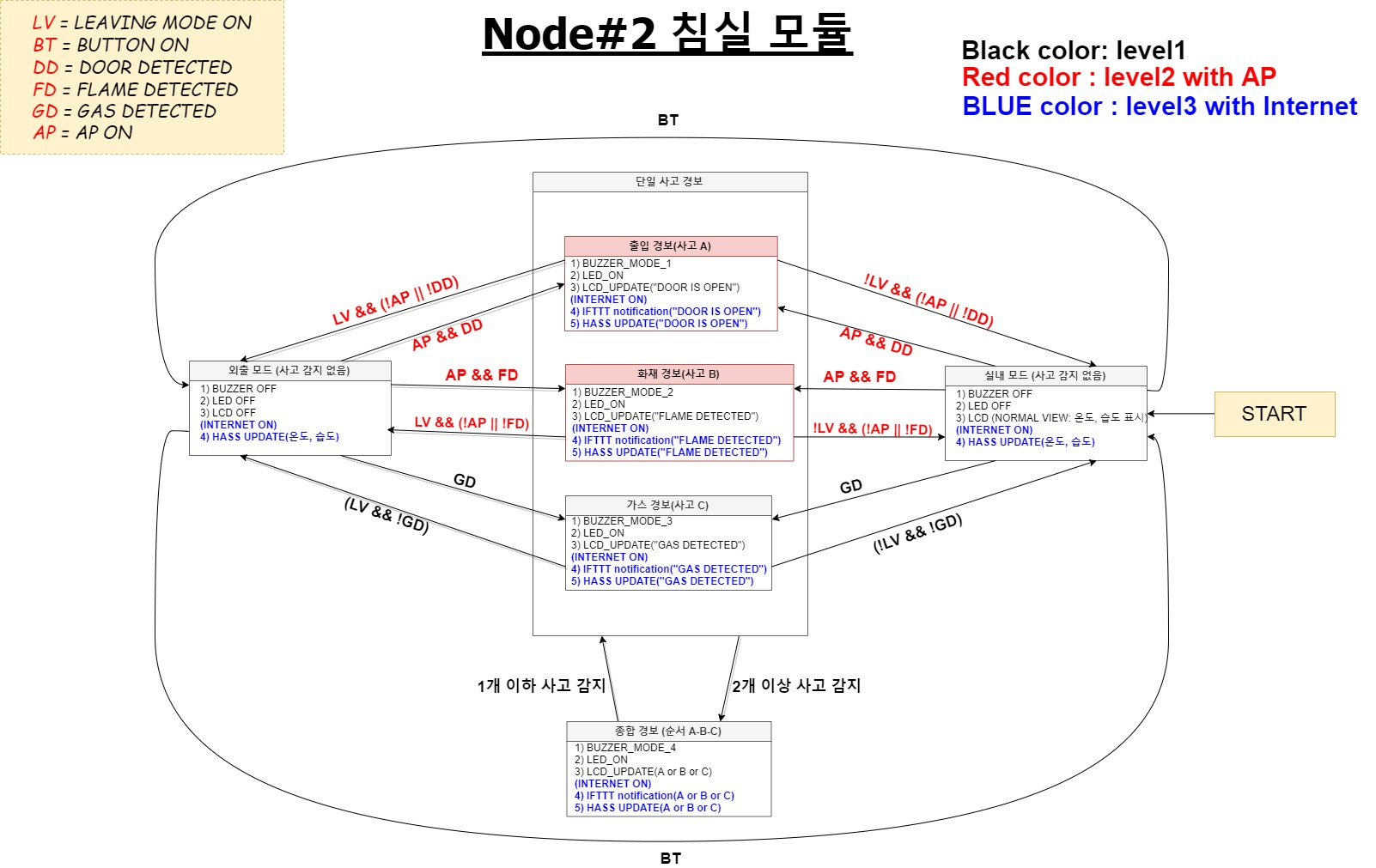
<그림 9> 문 열림 마그네틱 센서

출처: <https://www.aartech.ca/ams-10c-amseco-mini-surface-mount-contact-centre-leads.html>

문 열림 감지를 위해 사용하는 많은 종류의 마그네틱 센서가 있는데 그 중 ‘amseco’에서 만든 마그네틱 센서를 사용한다. 두 개의 선이 모듈에 연결되어 있는데 한 쪽을 GND에 연결하고 나머지 한쪽을 NodeMCU의 Digital pin과 연결한다. 마그네틱 센서가 붙어 있을 경우에는 선을 통해 LOW가 출력되며 문이 열린 상황, 즉 마그네틱 센서가 떨어지게 되면 선을 통해 HIGH가 출력된다.

* 1. Overall System Architecture





* 1. 기능 구현 및 설명
     1. 하드웨어

모듈은 크게 두 가지로 나누어진다. 첫 번째는 침실 모듈이고 두 번째는 현관 모듈이다. 침실 모듈을 구성하고 있는 구성 항목들은 다음과 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| 침실 모듈 | | | |
| 항목 | 개수 | 항목 | 개수 |
| WeMos D1 | 1 | LCD | 1 |
| LED | 1 | Buzzer | 1 |
| Button | 1 | BME280 | 1 |
| MQ-2 | 1 |  |  |

침실 모듈의 핀 구성은 다음과 같다.

D0 LCD-RST

D1 LED

D2 BUZZER

D3 BME-SDA

D4 BME-SCL

D5 LCD-CLK

D6 LCD-D/C

D7 LCD-DIN

D8 LCD-CS

TX(GPIO로 사용) BUTTON

A0(GPIO로 사용) MQ2

현관 모듈을 구성하고 있는 항목들은 다음과 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| 현관 모듈 | | | |
| 항목 | 개수 | 항목 | 개수 |
| WeMos D1 | 1 | LCD | 1 |
| LED | 1 | Buzzer | 1 |
| Button | 1 | 문열림감지센서 | 1 |
| 불꽃감지센서 | 1 |  |  |

현관 모듈의 핀 구성은 다음과 같다.

D0 LCD-RST

D1 LED

D2 Buzzer

D3 DoorLock

D4 불꽃감지

D5 LCD-CLK

D6 LCD-D/C

D7 LCD-DIN

D8 LCD-CS

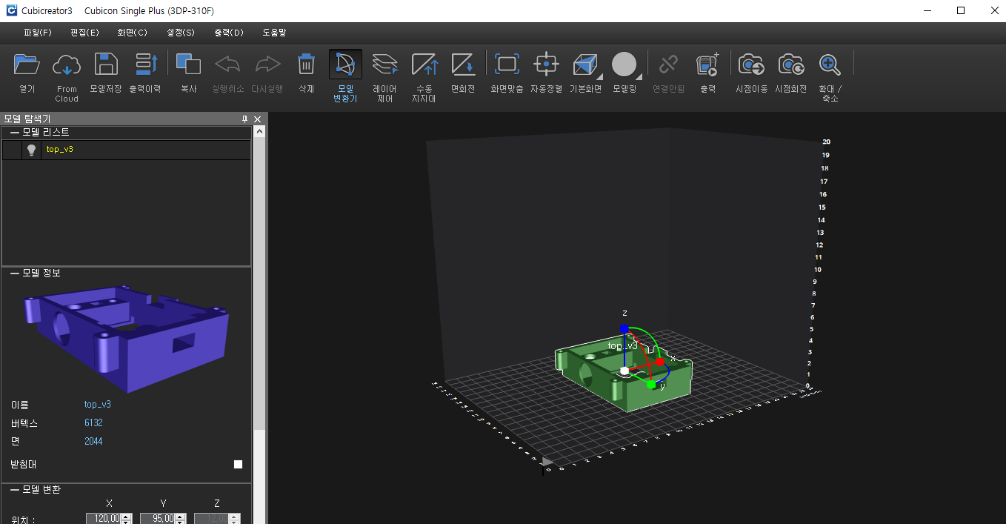
TX Button

각 모듈의 하드웨어를 설계하고 하나의 결과를 출력하는 과정은 총 3가지의 과정으로 이루어진다. 먼저 CATIA를 통해 필요한 모듈의 .stl파일을 제작한다. .stl파일이란 Stereo Lithography의 줄임말로 3D 모델링 된 데이터를 표준형식의 파일로 저장하는데 제공하는 파일 형식이다.



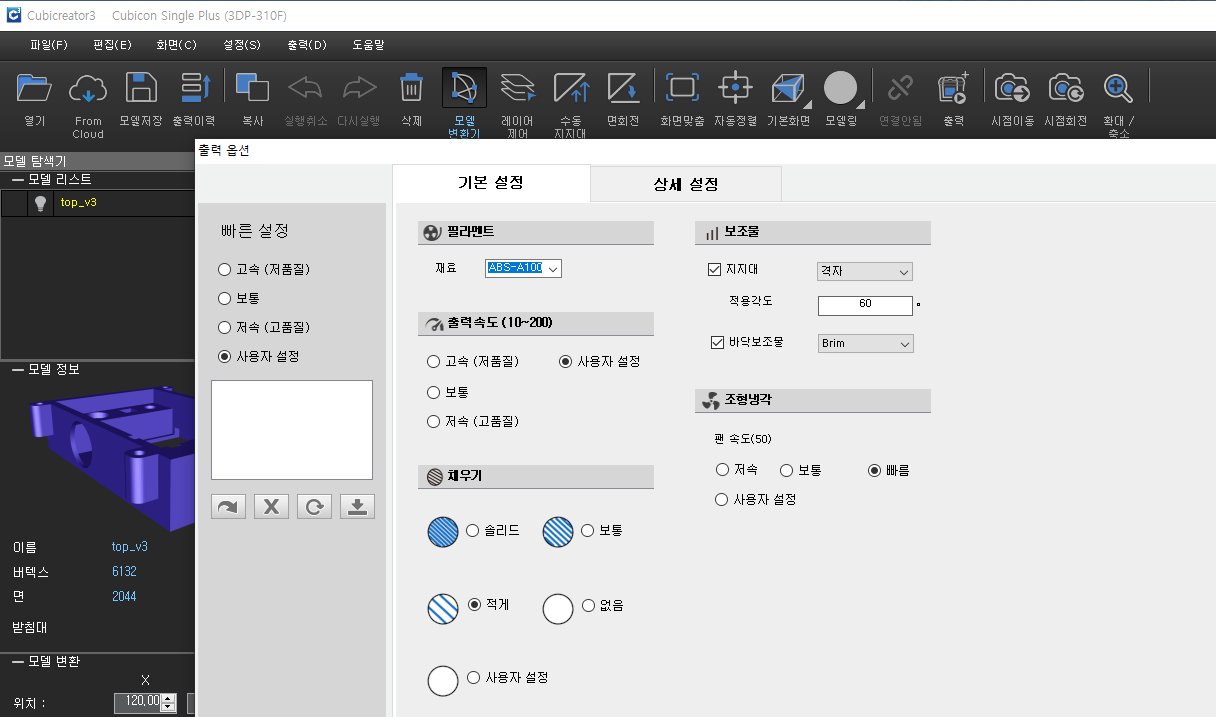
<그림> 왼쪽: 설계한 모듈 top cover, 오른쪽: top cover의 STL 파일

3d 모델링 툴인 CATIA를 이용하여 .stl 파일을 만들면 만든 파일을 3d printer로 뽑을 수 있도록 .gcode 파일으로 변환해줘야 한다. 변환해주는 툴에는 다양한 종류가 있지만, 우리 학부에서 사용하는 3d printer에서 제공하는 툴인 Cubicreator3를 사용한다. Cubicreator3에 .stl 파일을 불러오면 다음과 같은 결과가 나온다.



<그림> Cubicreator를 이용한 STL 파일 변환

이어서 출력버튼을 눌러 현재 우리가 사용하고 있는 필라멘트를 이용하여 printing 할 수 있도록 각 항목마다 최적의 값을 입력하고 출력 종류와 방법을 결정한다. 출력 종류와 방법에 따라 출력 시간이 달라지는데 우리는 printer를 한 대만 사용하므로 최대한 시간을 줄일 수 있도록 설정을 하였다.



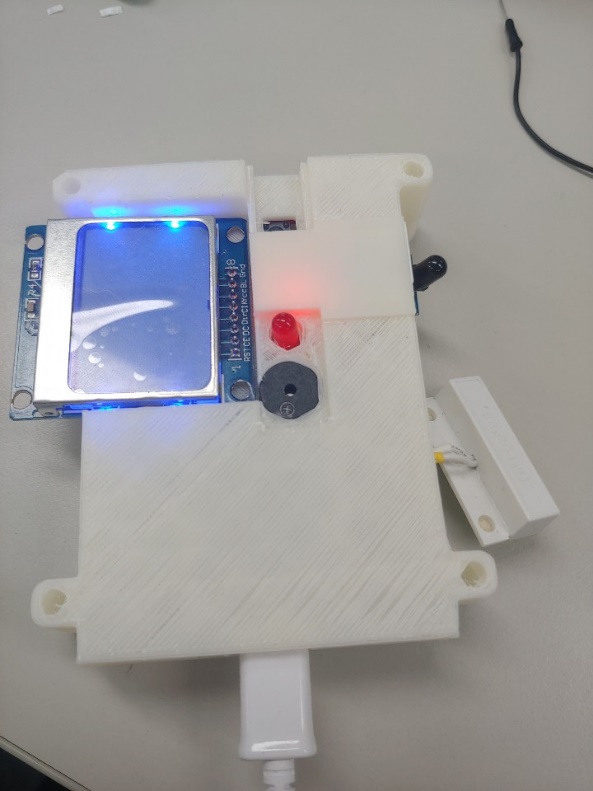
<그림> Cubicreator를 이용한 STL 파일 변환 설정

위의 과정을 거쳐 출력을 누르면 gcode로 구성된 .hfb 확장자 파일이 생성된다. 이 파일을 3d printer에 Wifi나 usb를 이용하여 옮기면 3d printer를 이용하여 출력하고자 했던 설계된 파일을 출력할 수 있다.

모듈 설명

#1 최종디자인- 현관모듈

박스형 디자인으로 외관을 구성할 것이었으므로 LCD를 장착할 공간을 확보하고, 부저와 LED는 외부에서 확인할 수 있게끔 설계하였다. 버튼은 사용자가 직접 누를 수 있게끔 디자인하였으며 불꽃감지센서는 측면으로 배치하여 센서가 회로에서 분리되지 않고, 돌출된 부분 없이 전체적인 높이가 일정하게끔하였다. 문열림 감지센서 역시 모듈외부에 있어야 하므로 회로와 센서를 연결하기위한 hole을 만들었다. USB케이블 삽입 시 WeMos보드가 위로 들리는 경우를 방지하기 위해 모듈 내부에 기둥을 배치하여 보드를 눌러주게끔 설계하였으며, USB케이블 삽입부의 두께를 프로토타입에 비해 감소시키고 위치또한 정확하게 설계하였다.



위의 설명처럼 전체적인 높이가 거의 일정하고, 부저와 LED는 외부에서 확인이 가능하며 LCD역시 사용자가 메시지를 직접 확인 할 수 있다. 문열림감지센서와 불꽃감지센서 역시 외부에 위치한다.

#2 최종디자인-침실모듈

외관이 박스형 디자인이므로 LCD를 장착할 공간을 확보하고, 부저와 LED는 외부에서 확인할 수 있게끔 설계하였다. 버튼은 사용자가 직접 누를 수 있게끔 디자인하였으며 온습도센서는 모듈 외부 상단부로 위치시켜서 발열이 있더라도 정상적으로 동작하게끔 하였으며 온습도센서가 흔들리거나 이동하면 안되기 때문에 감지부를 제외하고는 커버를 덮게끔 설계하였다. 가스센서는 하단부에 위치하는데 감지부를 외부로 돌출시키고, 흔들리지 않게 고정시키기 위하여 모듈의 내벽의 절반은 가스센서 모양인 사각형으로. 나머지 절반은 감지부인 원형으로 파여진 형태로 디자인하였다. 또한 USB케이블 삽입 시 WeMos보드가 위로 들리는 경우를 방지하기 위해 모듈 내부에 기둥을 배치하여 보드를 눌러주게끔 설계하였으며, USB케이블 삽입부의 두께를 프로토타입에 비해 감소시키고 위치 또한 정확하게 설계하였다.



BME280센서는 상단부에 위치 및 고정되어 있으며 발열에 큰 영향을 받지 않는다. 하단에 연결된 MQ-2센서 역시 고정되어있으며 사용자가 LCD에 출력되는 메시지를 확인하고, LED및 부저 알람을 확인 할 수 있다.

* + 1. 소프트웨어
       1. Level1

2.3.2.1.1 Level 1 상황 정의

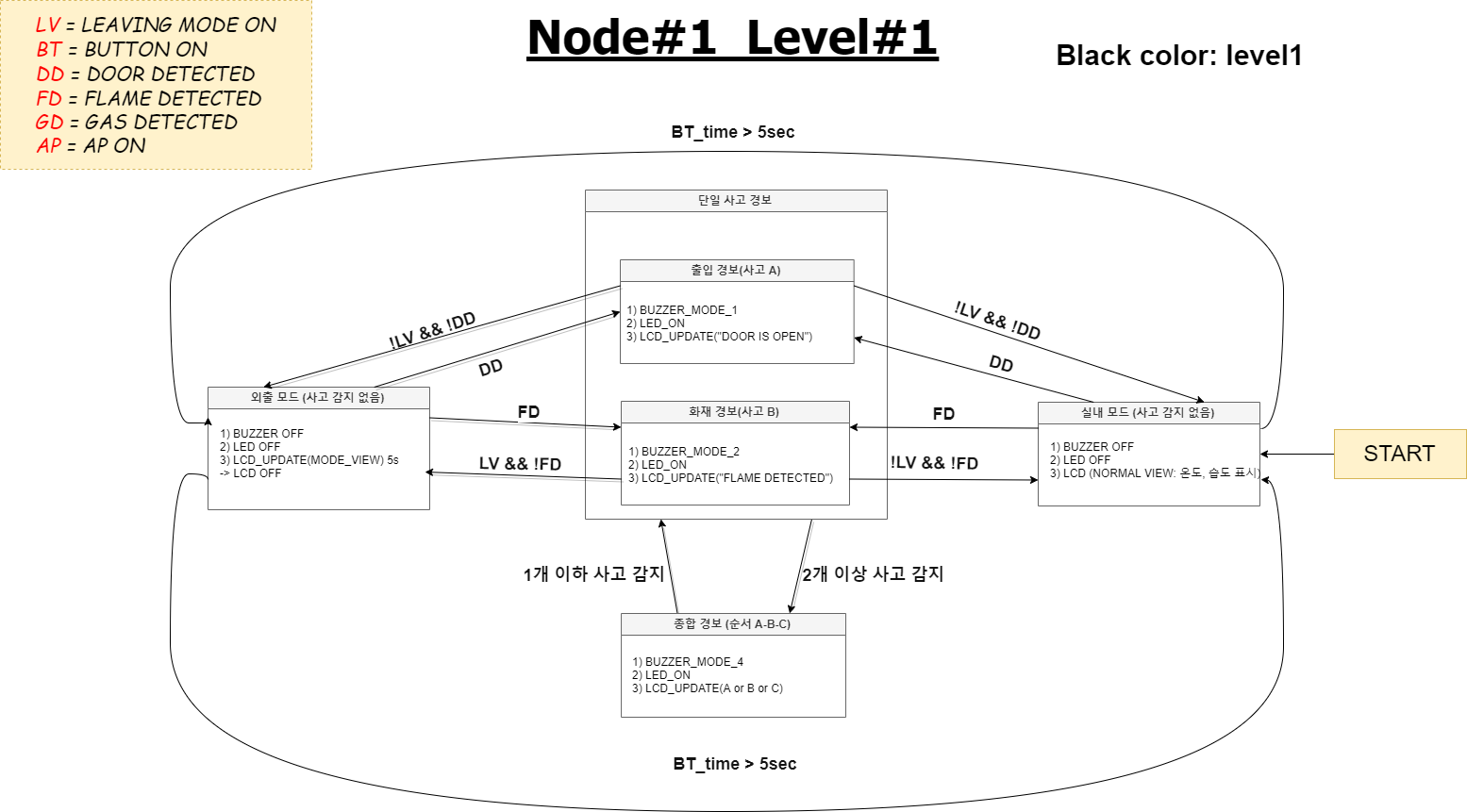
본 단계(level 1)에서는 외부와의 통신(엑세스 포인트 기반 통신 및 인터넷 연결 등)이 이루어지지 않은 채로 Node1과 Node2가 독립적으로 각각 내부 컴포넌트들에 의한 상호작용만이 이루어 진다.

2.3.2.1.2 Node1에서의 구현

하드웨어 구현 과정에서 언급한 바와 같이, Node 1를 구성하는 컴포넌트는 문열림 감지 센서, 화재 감지 센서, 버튼, LED, Buzzer, 그리고 LCD이다. 본 단계에서는 상황에 따라 가스 경고 모드, 실내 모드, 외출 모드, 위 3가지 모드의 변환이 이루어진다. 모듈의 주요 기능은 아래와 같다

1. 초기모드는 실내모드이고, 만약 버튼이 5초간 클릭될 경우 외출모드<->실내모드 변환이 이루어진다.
2. 만약 버튼을 5초간 클릭될 경우 외출모드<->실내모드 변환이 이루어진다.
3. 화재감지는 외출모드, 실내모드 상관없이 감지되었을 때, 부저, LED, LCD 화면이 출력되고 감지가 안되면 부저, LED, LCD 화면이 정상으로 돌아 온다.
4. 문열림 감지는 실내모드일 때, 감지되면 부저, LED, LCD 화면이 출력되고 감지가 안되면 부저, LED, LCD 화면이 정상으로 돌아 온다. 외출 모드 감지되면 부저, LED, LCD 화면이 출력되고 감지가 안된경우에도(즉 문이 닫힌경우)에도 사용자가 와서 버튼을 누르기 전까지 계속 울린다.

위 4가지 기능 상에서 나타난 모드 변환의 과정은 아래 상태 변화도에 나타난다.



<그림. Node#1 상태 변화도(level 1 단계)>

BUZZER\_MODE\_1, BUZZER\_MODE\_2, BUZZER\_MODE\_3는 상황에 따라 다른 소리를 내기 위해 구분하였다.

|  |  |  |
| --- | --- | --- |
| BUZZER\_MODE | 실내모드 | 외출모드 |
| BUZZER\_MODE\_1 | tone(Buzzer, 262, 100); | tone(Buzzer, 262, 100); |
| BUZZER\_MODE\_2 | tone(Buzzer, 131, 100); tone(Buzzer, 262, 100);  0.5초 주기로 5초간 동안 | tone(Buzzer, 131, 100); tone(Buzzer, 262, 100);  0.5초 주기로 10초간 동안 |
| BUZZER\_MODE\_3 | tone(Buzzer, 131, 100); tone(Buzzer, 262, 100);  0.2초 주기로 5초간 동안 | tone(Buzzer, 131, 100); tone(Buzzer, 262, 100);  0.2초 주기로 10초간 동안 |

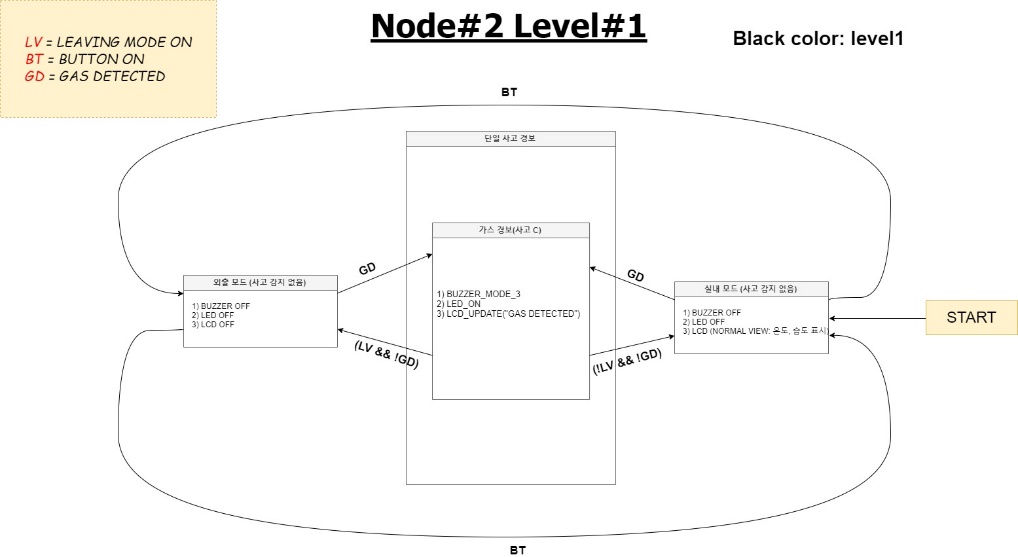
<그림. Node#1 상태에 따른 buzzer mode(level 1 단계)>

2.3.2.1.3 Node2에서의 구현

하드웨어 구현 과정에서 언급한 바와 같이, Node 2를 구성하는 컴포넌트는 가스센서, 온습도센서, 버튼, LED, Buzzer, 그리고 LCD이다. 본 단계에서는 상황에 따라 가스 경고 모드, 실내 모드, 외출 모드, 위 3가지 모드의 변환이 이루어진다. 모듈의 주요 기능은 아래와 같다.

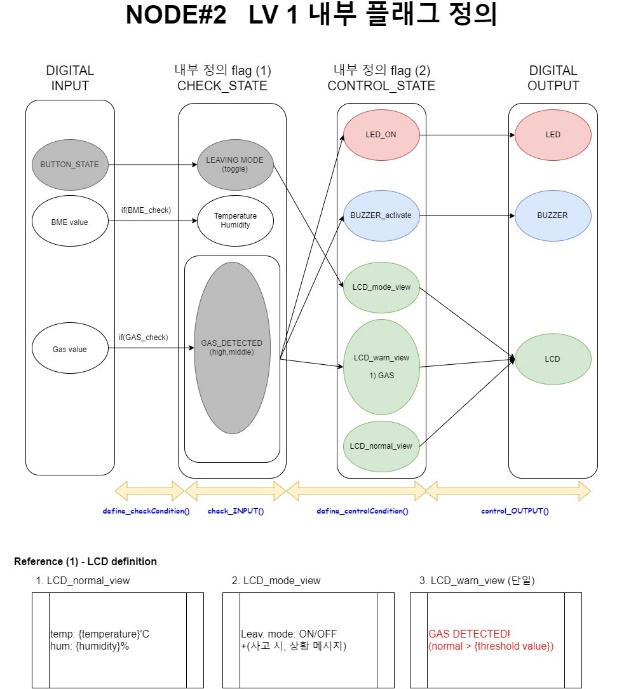
1. 초기모드는 실내모드이고, 만약 버튼이 클릭될 경우 외출모드<->실내모드 변환이 이루어진다.
2. 온습도를 실내모드 상에서는 5분, 외출모드 상에서는 15분마다 BME를 통해 측정한다. 실내 모드일 경우, LCD는 측정한 온습도는 화면 상에 출력한다.
3. 가스 농도를 어떤 모드인지 관계없이 1초마다 측정한다. 그 후, 가스농도가 기준 수치(250:중간 수준, 400:높은 수준)를 넘기면 가스 감지 경고 모드로 변환한다.
4. 가스 경고 모드 상에서 가스가 기준치보다 내려갔을 경우, 해당 모듈의 기존 모드(실내/외출 모드)를 판별하여 해당 모드로 다시 변환한다.

위 4가지 기능 상에서 나타난 모드 변환의 과정은 아래 상태 변화도에 나타난다.



<그림. Node#2 상태 변화도(level 1 단계)>

위 모드 변환을 소프트웨어 상에서 구현할 때, 이를 주관하는 주요 제어 플래그들은 상태도에 나타난 바와 같이 LV(외출 모드인지), BT(해당 시점에 버튼이 눌렸는지), GD(가스농도가 기준치를 넘겼는지), 이렇게 총 3가지이다. 위 내부 변수들의 상태 변화에 따라 모드가 변화하고, 그에 따라 나타나는 기능도 달라지게 된다. 내부 플래그가 프로그램 상에서 정의되고 변환되는 흐름을 나타낸 상태표는 아래와 같다. (LCD 작동 명세 포함)



<그림. Node#2 내부 플래그 변화도(level 1 단계)>

검은색 음영으로 칠해진 부분이 상태 변화도에 나타난 3가지 주요 내부 플래그들을 의미한다. 위 플래그 변화도 상의 BUTTON\_STATE는 상태 변화도 상의 BT를 나타내고, 디지털 입력 단계에서 실제로 버튼이 입력이 되었는지를 나타낸다. 마찬가지로 LEAVING MODE는 LV를 나타내며, 현재 실내모드인지를 나타낸다. 마지막으로 GAS\_DETECTED는 GD를 나타내며, 입력 단계에서 가스를 검출하여 기준 수치를 넘겼는지 확인하는 플래그이다. 위 단계가 프로그램 상에서 define\_checkCondition()과 check\_INPUT() 두 함수를 통해 이루어 진다. 디지털 입력 단계를 거쳐 위 플래그들(이하 check 플래그)을 변화시킨 후, 그에 따른 디지털 출력을 정의하기 위해, 각 컴포넌트(LED, Buzzer, LCD)의 기능을 관장하는 또다른 내부 플래그들이 존재한다. 각 컴포넌트 출력의 상태를 명시하는 해당 플래그(이하 control 플래그)들은 check 플래그들의 상태 변화에 의해 정의되며, 그에 따라 디지털 출력을 결정한다. LED 관련 플래그는 ON/OFF 두 가지 상태만을 나타낸다. Buzzer 관련 플래그는 위 단계는 프로그램 상에서 define\_controlCondition()과 control\_OUTPUT() 두 함수를 통해 이루어진다.

정리하자면 다음과 같다. level1(node#2) 단계에서의 코드는 총 4가지 단계의 함수로 구성된다. 먼저 실행되는 두 함수는 디지털 입력과 그에 대응되는 check 플래그들을 정의한다. 그 다음 실행되는 두 함수는 check 플래그들에 대응되는 control 플래그와, 그에 따른 디지털 출력을 실행한다. 해당 프로그램은 non-blocked하게 진행되며, check 플래그와 control 플래그를 매 loop마다 재정의하여 그에 따른 출력을 결정한다.

* + - 1. Level2
         1. Level2 상황 정의

레벨 2 는 단독으로 동작하는 레벨 1 과 달리 AP 가 추가됩니다. 그리고 그 AP 를 통하여 노드들이 서로의 센싱 상황을 공유할 수 있게 됩니다. AP 가 추가되었지만 중앙서버로 사용할 수 있는 기기는 없습니다. 그렇기 때문에 각 노드들을 서버로 만들어서 센싱 상황을 공유받 을 수 있게 해야 합니다.

* + - * 1. Level2 기능 구현

|  |  |
| --- | --- |
| 센서 | 변수 |
| 문 열림 감지 센서 | #define WEBHOOK\_PATH\_DOOROPEN WEBHOOK\_PROTOCOL NODE2\_IP "**/DoorOpen/**" |
| 불꽃 감지 센서 | #define WEBHOOK\_PATH\_FLAME WEBHOOK\_PROTOCOL NODE2\_IP "**/Flame/**" |
| 가스 센서 | #define WEBHOOK\_PATH\_GAS WEBHOOK\_PROTOCOL NODE1\_IP "**/Gas/**" |
| 온도 센서 | #define WEBHOOK\_PATH\_TEMP WEBHOOK\_PROTOCOL NODE1\_IP "**/Temperature/**" |
| 습도 센서 | #define WEBHOOK\_PATH\_HUMIDITY WEBHOOK\_PROTOCOL NODE1\_IP "**/Humidity/”** |

이러한 상황 속에서 기능을 구현하기 위해 가장 간단하게 구축할 수 있는 서버인 웹서버를 구축하여 센싱 상황을 공유할 수 있게 하였습니다. 그리고 약속된 경로에 HTTP 프로토콜의 GET 패킷을 보내면 특정 상황이 발생하였음을 알릴 수 있도록 하였습니다. 약속된 경로는 위에 표입니다. WEBHOOK\_PROTOCOL은 "http://"이고, NODE1\_IP와 NODE2\_IP 각각 모듈의 IP로 정의되어 있습니다. 즉 모든 사고 상황들에 대하여 웹서버의 경로를 약속해두고 사고를 감지하였을 때 상대 노드의 웹서버 링크에 HTTP GET 패킷을 전송하여 사고 상황을 공유하게 하였습니다. 또한 이렇게 HTTP 호출을 통하여 특정한 동작을 약속하는 것을 웹훅(Webhook) 이라 하기 때문에 앞으로 웹훅이라는 용어를 사용하여 설명을 이어가겠습니다.

* + - * 1. 웹훅을 사용한 Node1 의 Level 2 구현

Node1 에서의 레벨 2 구현은 다음과 같습니다. 먼저 Node 1 이 사고 상황을 감지하면 Node 2 에서 미리 정의된 링크를 호출합니다. Node 1 은 화재상황과 문열림사고를 감지할 수 있기 때문에 Node 2 의 웹서버가 “/Flame” 과 “/DoorOpen” 이라는 미리 약속된 링크로 HTTP 호출이 수신되는지 대기하고 있습니다. 그러면 Node 1 에서 화재상황이나 문열림사고가 발생하면 각각 “http://NODE2\_IP/Flame”또는 “http://NODE2\_IP/DoorOpen”으로 HTTP 호출을 합니다.

* + - * 1. 웹훅을 사용한 Node2 의 Level 2 구현

Node2 에서의 레벨 2 구현은 Node 1 의 경우와 비슷합니다. Node 2 이 사고 상황을 감지했을 때 Node 1 에서 미리 정의된 링크를 호출합니다. Node 2 은 가스 누출 상황과 온습도를 감지할 수 있기 때문에 Node 1 의 웹서버가 “/Gas” , “/Humidity”, “/Temperature” 라고 미리 약속된 링크로 HTTP 호출이 수신되는지 대기합니다. 그러면 Node 2 에서 가스 누출 사고가 발생하면 “http://NODE1\_IP/Gas” 로 HTTP 호출을 합니다. 하지만 온습도 정보는 사고 상황과 관계 없이 항상 전송해야 하므로 “http://NODE1\_IP/ Humidity/{습도}”, “http://NODE1\_IP/Temperature/{온도}”의경로로 항시 HTTP 호출을 합니다. 그러면 Node2 에서 이 HTTP 호출을 수신하고 온습도 정보를 알 수 있게 됩니다.

* + - 1. Level3

2.3.2.3.1 Level3 상황 정의

레벨 3는 레벨2에 이어서 인터넷이 가능한 상황이 된다. 이때 이어서 구현하는 기능은 두가지로 사고가 감지될 경우 각 노드에서 현재상황을 MQTT를 통해 HASS에 전달하여 사용자가 집안에 없더라도 현재 사고 상황을 바로 확인할 수 있는것과 사고가 일어날 시 사용자의 IFTTT앱을 통해 사고 상황을 알 수 있도록 한다.

* + - * 1. Level3 기능 구현

Level3에서 사용자에게 사고 상황을 전파하기 위해 IFTTT앱을 사용하여 사고가 감지되면 IFTTT앱을 통해 사용자에게 알림이 가도록 한다. IFTTT앱으로 사고 상황을 전당하기 위해서 사용한 함수는 다음과 같다.

|  |
| --- |
| void send\_event(const char \*event){  WiFiClient client;  const int httpPort = 80;  if (!client.connect(IFTTT\_HOST, httpPort)) {  return;  }  String url = "/trigger/";  url += event;  url += "/with/key/";  url += IFTTT\_KEY;  String mJson = String("{\"value1\": \"") + "IoT System Design... \"}";  client.println(String("POST ") + url + " HTTP/1.1");  client.println(String("Host: ") + IFTTT\_HOST);  client.println("Content-Type: application/json");  client.print("Content-Length: ");  client.println(mJson.length());  client.println();  client.println(mJson);  delay(300);  client.stop();  } |

<사진@#$> send\_event 함수

사고가 감지되면 send\_event 함수가 동작하는데 이 때 사고에 대한 정보가 IFTTT로 전달된다. IFTTT에서 특정 사고에 대한 정보가 들어왔을 때 알림이 오도록 미리 설정해 놓으면 사고에 대한 정보가 들어왔을 때에 사용자의 앱을 통해 알림이 온다. 전체 동작 원리는 다음과 같다.

‘send\_event’ 함수가 실행이 되면 가장 먼저 maker.ifttt.com의 80번 포트에 연결을 시도한다. 만약 연결이 되지 않을 경우에는 다시 loop의 처음으로 돌아가 시작된다. Server와 연결되면 데이터(사고 정보)를 보내기 위해 POST web request를 사용하여 IFTTT에 보낸다. 데이터를 POST하기 위해서 “http://maker.ifttt.com/trigger/일어난 event의 이름/with/key/IFTTT의 고유키” url에 접속하여 post한다. 사용자가 사전에 등록한 데이터가 url에 post될 경우에 알림이 오도록 설정해 놓으면 사고 시 사용자의 앱으로 알림이 온다. 각 사고 상황에 대한 event의 이름은 다음과 같다.

|  |  |
| --- | --- |
| 센서 | Event이름 |
| 문 열림 감지 | DoorFlagon |
| 화재 감지 | FlameFlagon |
| 가스 감지 | GasFlagon |

<사진> 사고 상황에 대한 event의 이름

인터넷과 연결이 되면 다음으로 하는 일이 HASS의 각 센서의 해당 주소에 메시지를 보내어 현재 상황을 홈페이지로 보여주는 것이다. mqtt통신을 하기 위해서 broker인 raspberry pi와 연결하였다.

|  |
| --- |
| mqtt\_client.setServer(mqtt\_broker, 1883);  mqtt\_client.setCallback(callback);  //Connection to MQTT broker  mqtt\_client.connect(mqtt\_nodeName, mqtt\_user, mqtt\_pwd); |

<사진>mqtt\_연결

Mqtt\_broker는 raspberry pi의 IP주소이다. MQTT통신으로 브로커에 메시지를 뿌려주고 라즈베리파이에서 HASS로 전송하는 과정 중에서 모듈에서 메시지를 보내는 방식은 다음과 같다. 다음 예제는 온습도 센서의 MQTT통신을 예로 들었다.

|  |
| --- |
| char data[80];  String payload\_dht = "{";  payload\_dht += "Hum: ";  payload\_dht += humidity;  payload\_dht += ", ";  payload\_dht += "Temp: ";  payload\_dht += temperature;  payload\_dht += "C";  payload\_dht += "}";  payload\_dht.toCharArray(data, (payload\_dht.length() + 1));  mqtt\_client.publish(pub\_dht22, data, 1); |

<사진> 모듈2에서의 온습도센서 mqtt통신

가장 먼저 String으로 변수를 설정한 다음 해당 센서를 통해 들어오는 값을 가지고 온다. 그 string값을 character 배열인 data에 올려준 다음 그 값을 HASS의 해당 topic에 메시지로 보내준다. pub\_dht22가 온습도센서의 온습도를 메시지로 주는 topic주소이다.

각 센서의 topic과 메시지는 다음과 같다.

|  |  |  |
| --- | --- | --- |
| 센서 | 메시지 | Topic |
| 문 열림 감지 센서 | “DOOR open!” (감지 되었을 때)  “DOOR closed” (감지되지 않았을 때) | "myhome/room1/door" |
| 불꽃 감지 센서 | “FLAME detected” (감지 되었을 때)  “FLAME undetected” (감지되지 않았을 때) | "myhome/room1/flame" |
| 가스 센서 | “Gas detected” (감지 되었을 때)  “Gas undetected” 감지되지 않았을 때) | "myhome/room2/gas" |
| 온습도 센서 (온도) | Temperature (항상) | "myhome/room2/dht\_t |
| 온습도 센서 (습도) | Humidity (항상) | "myhome/room2/dht\_h" |
| 온습도 센서 (온습도) | “{“+ Hum: “ + humidity +”,” + “Temp: ”+ temperature +”C” +” }”  (항상) | “myhome/room2/dht22” |
| Wifi (거실모듈) | “WIFI ON” (wifi가 켜졌을 때)  “WIFI OFF” (wifi가 꺼졌을 때) | “myhome/room1/wifi” |
| WIFI (현관모듈) | “WIFI ON” (wifi가 켜졌을 때)  “WIFI OFF” (wifi가 꺼졌을 때) | “myhome/room2/wifi2” |

<사진> 각 센서의 topic과 메시지

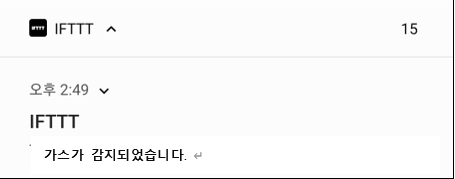
HASS는 라즈베리파이를 통해 동작한다. 라즈베리파이가 각 모듈에서의 해당 센서들의 topic을 subscribe한 다음 값이 올때마다 화면에 뿌려주었다. HASS 홈페이지는 다음과 같은 group.yaml 파일로 나누어 주었다.

|  |
| --- |
| default\_view:    name: "temperature and humidity"    entities:      - sensor.temp\_humidity      - sensor.Temperature      - sensor.Humidity  Wifi:    name: "WiFi state"    entities:      - sensor.Wifi1      - sensor.Wifi2    Room1:    name: Room1    entities:      - sensor.Door      - sensor.Flame    Room2:    name: Room2    entities:      - sensor.Gas    Room1\_menu:    view: yes    name: Room1    entities:      - group.room1    Room2\_menu:    view: yes    name: Room2    entities:      - group.room2 |

Wifi는 현관모듈과 거실 모듈의 wifi 상황을 알려주고 각각 모듈의 센서들이 현재 어떤 상황에 있는지 묶어 볼 수 있게 하였다. Room1이 거실모듈의 문열림감지센서와 불꽃감지센서의 상태를 보여주고, Room2가 가스의 현재 상태를 보여준다. 각각 모듈에서 보내주는 메시지와 동일한 형태로 출력이 되는 것을 볼 수 있다.

* 1. 최종 시뮬레이션 결과
     1. IFTTT앱

Level3의 결과물은 다음과 같다.



1. <그림> IFTTT notification

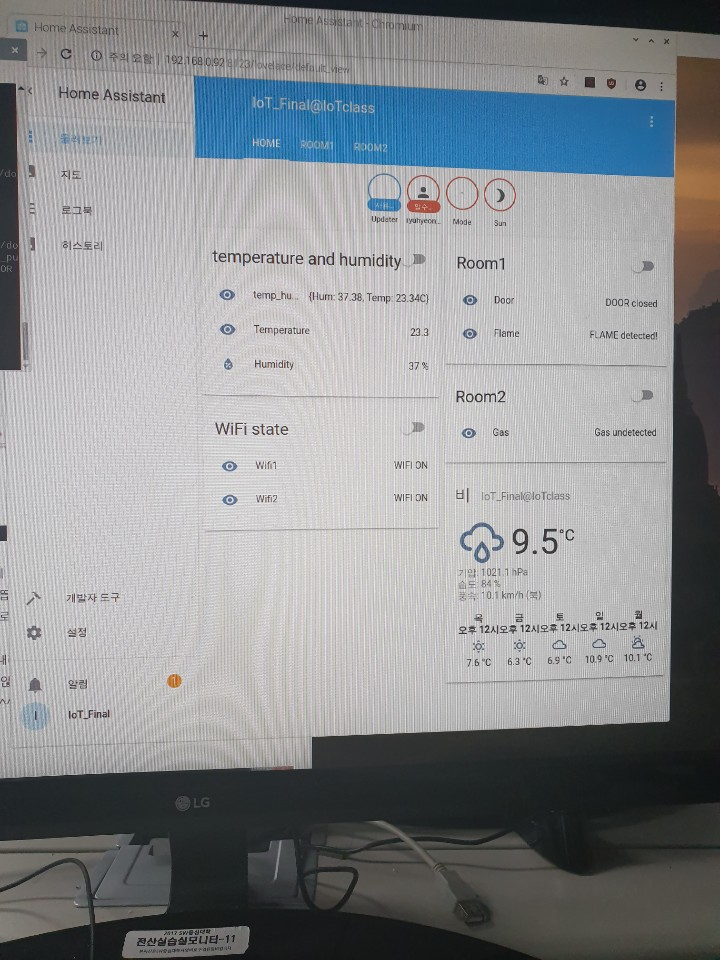
<그림>는 가스가 감지되었을 때 IFTTT 어플리케이션을 통해 온 알림이다.

|  |  |
| --- | --- |
| **센서** | **IFTTT앱 메시지** |
| 문 열림 감지 | 문열림이 감지되었습니다 |
| 화재 감지 | 화재가 감지되었습니다. |
| 가스 감지 | 가스가 감지되었습니다. |

<사진> IFTTT알림 메시지

2.4.2 HASS 결과물

mqtt통신을 해서 나온 HASS 홈페이지는 다음과 같다.



<사진> 화재 감지되었을 때 HASS화면

다음 화면은 화재가 감지되었을 때의 HASS화면이다. 각 모듈에서 보내주는 data(message) 그대로를 라즈베리파이에서 바로 받아와 홈페이지에 올려준다.

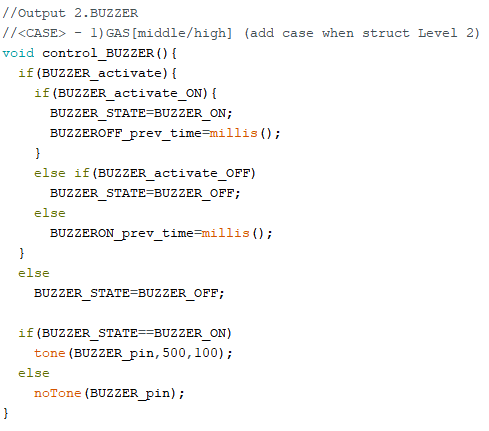
* 1. 프로젝트 중 발생한 문제점과 문제 해결 과정

3.1.1 함수 최적화

현관 모듈과 거실 모듈의 level1에서의 작동은 문제가 되지 않았지만 와이파이와 연결하면서 여러가지 기능이 추가적으로 이어졌을 때 컴포넌트들이 원하는 대로 동작하지 않았다.

3.1.2 clock 작동 간의 부조화 node#2

Node#2 코드 상에서 level 2 이상으로 넘어갔을 경우, 부저가 제대로 작동하지 않는 문제가 발생하였습니다. Level 1에서는 정상적으로 작동하는 것으로 보아 부저 작동 함수의 논리적 구조의 문제라기 보다는 시스템 clock 작동 간의 부조화로 인해 나타나는 문제인 것으로 판단했습니다. 기존의 부저 작동 함수와 수정한 함수는 아래와 같습니다.



좌측의 수정 전 함수는 부저가 작동할 경우 activate state를 적용하였고, 부가적으로하위STATE인 activate\_on/off 를 활용하여, 부저가 toggle 되도록 하였습니다. 위와 같은 구조는 논리적으로는 문제가 없었기에 level 1에서는 정상적으로 작동하였지만, level 2이상에서 통신 기능이 추가될 경우 전체 코드의 작동 시간이 길어져 STATE 간의 충돌, 혹은 최신화가 늦어지는 현상이 발생하는 것으로 실험을 통해 판단하였습니다. 따라서 최소한의 STATE를 사용하여 작동시간을 최대한 줄이고 충돌을 방지하기 위해 부저 toggle 기능을 제하고 각 사고 상황마다 다른 음을 출력하여 구별하도록 수정하였습니다. 기존의 프로그램은 부저 toggle 속도로 사고 상태를 구분했다면, 수정 후의 프로그램은 부저 음의 높낮이로 사고 상태를 구분하게 되었습니다.

3.1.3 코드 모듈화 및 clock 작동 간의 부조화 node#1

Node#1 코드 상에서 level 2 이상으로 넘어갔을 경우, 부저가 제대로 작동하지 않는 문제가 발생하였습니다. Level 1에서는 정상적으로 작동하는 것으로 보아 부저 작동 함수의 논리적 구조의 문제라기 보다는 이것 또한 시스템 clock 작동 간의 부조화로 인해 나타나는 문제인 것으로 판단했습니다. Node#2처럼 모듈화를 하여 상태 파악하는 곳, 액션을 결정하는 곳, 액션을 취하는 곳으로 나눴다면 코드 보기에도 활용하기에도 더 잘 되었을 것 같습니다. 모듈화 하지 않아서 생기는 문제 다음에 clock 작동 간의 부조화로 더 해결 방안은 interrupt 함수를 이용해서 buzzer 문제를 해결하는 방안을 생각했습니다.

1. 결론

최종 결과물인 무선종합경보기의 전체적인 구성은 2개의 모듈로 구성된다. 불꽃감지센서, 문열림감지센서, LCD가 있는 현관모듈과 가스모듈, 온습도 센서, LCD가 있는 거실모듈 총 2개의 모듈로 구성된다. 각 모듈은 세가지 레벨로 구성되어 있고 각 레벨은 다음과 같은 상황에 대한 대처 방법이다. 첫번째 레벨은 인터넷과 AP가 둘 다 없는 상태를 나타내고 두번째 레벨은 인터넷은 없지만 AP는 있는 상황, 마지막으로 레벨3은 AP와 인터넷 모두가 있는 상황이다. 레벨1 상황에서의 각 모듈들은 탑재되어 있는 센서들을 통해 사고를 감지하고 사고가 감지되면 LED와 Buzzer를 통해 사람들이 사고 상황에 대해서 인지할 수 있도록 신호를 출력한다. 하지만 다른 모듈의 사고까지는 감지를 하지 못한다. 레벨 2는 레벨1에 더해져 다른 모듈의 사고까지 인지하고 LCD를 통해 어떤 사고가 나고 있는지 사용자에게 알려준다. 레벨3은 각각의 모듈이 통신을 하고 또한 IFTTT앱을 통해 등록되어 있는 사용자의 application을 통해 사고 상황에 대한 알림을 보낸다. 또한 각각의 모듈이 라즈베이파이로 mqtt통신을 하여 현재 어떤 상황에 있는지 사용자가 밖에 있어도 사고를 감지 및 상황에 대한 정보를 HASS를 통해서 볼 수 있다.

1. 부록
   1. Individual Project Assignment

태연

1.#2침실모듈 회로 구상 및 연결

2.회로 연결 후 정상적으로 동작하는지 확인하기 위해 온습도센서와 버튼용 테스트코드 각각 작성

3.#2침실모듈의 초기디자인, 프로토타입 디자인, 최종모듈 디자인 구상

4.#1현관모듈의 최종모듈 디자인구상에 일조

5.프로젝트 진행중 하드웨어에 문제가 있는 경우 문제 확인 및 수정

6.소프트웨어 담당 팀에게 하드웨어 관련 수정이유 및 수정사항 정보고지

7.보고서 2.3 기능 구현 및 설명 파트 하드웨어부분 기능설명 부분 작성

8.보고서 2.5 프로젝트 중 발생한 문제점과 문제 해결 과정 하드웨어부분 작성

민수

1. 현관모듈 회로 구성 및 전체 설계

2. 침실모듈 회로 구성 및 설계 피드백

3. 3d 모델링 툴을 이용한 현관모듈과 침실모듈 3d 모델 설계 및 제작

4. 보고서 개요 구성 및 보고서 작성 (서론 및 하드웨어 기능 구현)

현아

1.소프트웨어 level3 코딩

2.레벨2 코딩 보조 및 피드백

3.보고서 작성 (level3부분)

찬솔

1. 소프트웨어 level2 코딩 총괄

2. 소프트웨어 level2 보고서 작성

3. 소프트웨어 level3 테스트, 수정보완, 및 마무리

경민

1. 소프트웨어 node#2 level1 코딩 총괄

2. 소프트웨어 node#2 프로그램 구조 및 상태표 도식화

3. 보고서 본론 부분 담당 영역 작성

아름

1. 소프트웨어 node#1 문열림, led, lcd 코딩

2. 소프트웨어 node#1 level2 코딩

3. 소프트웨어 level3 보조 및 피드백

4. 보고서 작성

5. node#1 상태표 도식화

교찬

1. 소프트웨어 node#1 코딩 총괄