아두이노

지각자 체크기

제작 기간 2019년 03월 16일 ~ 2019년 07월 12일
제출 기간 2019년 07월 12일

제작 동기

지각생에게 지각비를 걷는 규칙을 제정하였고, 학급 총무가 지각자를 일일이 체크 하는 과정에서 총무와 친한 친구들이 특혜를 받는 문제점이 생김

제작 과정

지각 문제를 자율 동아리 주제로 선정

- 참여인원 : 9명 (팀장 심재빈)
- 제작기간 : 4개월
- 제작언어 : C언어
- 제작장비 : 아두이노

결과

지각자 체크기를 제작, 교내 학술탐구대회에서 발표하여 최우수상으로 입상

나의 변화

부원들의 부족한 역량으로 인해 소외되는 문제점을 배려와 협력을 통해 극복할 수 있다는 점을 깨닫게 됨

보고서 작성 · 심재빈

팀 발표자료

학급의 지각자 체크를 위해 지하철 게이트를 모티브로 하여 아두이노와 C언어 프로그래밍을 이용해 학급에 최적화된 게이트를 제작함

1. 탐구 동기 및 목적

학급 규칙으로 지각한 학생에게 '지각비'를 걷는 규칙을 제정하였는데, 학급 총무가 지각자를 일일이 체크해야 한다는 불편함과 총무와 친한 친구들이 특혜를 받게 되는 문제점이 있었다.
문제를 해결하기 위해 지하철의 출입 게이트를 모티브로 하여 개인의 고유 식별 카드(RFID)를 태그함으로써 각자의 등교 시간을 확인하고 자동으로 기록되는 학급에 최적화된 게이트를 만들게 되었다.

2. 선행연구

■ RFID (RADIO-FREQUENCY INDENTIFICATION)

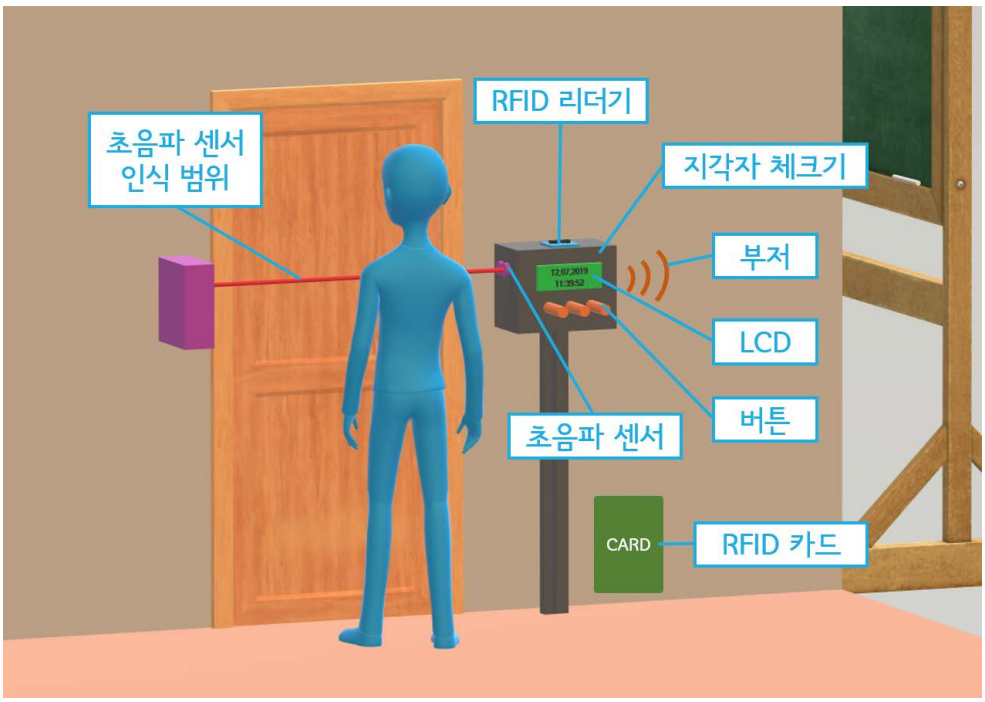
주파수를 통해 개체를 식별하는 방식의 기술로, 전파를 이용하여 직접 접촉하지 않더라도 먼 거리에서 정보의 식별이 가능한 것이 특징이다. RFID 태그는 집적 회로 속에 저장된 정보를 안테나를 통해 송신하고, 이를 통해 RFID 태그가 부착된 대상을 식별할 수 있다.

■ 아두이노

다양한 스위치나 센서로부터 입력 값을 받아들여 LED 나 모터와 같은 전자 장치들로 출력을 제어함으로써 환경과 상호작용이 가능한 물건을 만들어 낼 수 있다. 예를 들어 단순한 로봇, 온습도계, 동작 감지기, 음악 및 사운드 장치, 스마트 홈 구현, 유아 장난감 및 로봇 교육 프로그램 등의 다양한 제품들이 아두이노를 기반으로 개발 가능하다.
아두이노의 소스코드는 C++ 언어를 기반으로 하기 때문에 C 언어의 표준라이브러리 함수를 이용하여 개발한다.

3. 설계

■ 체크기 동작

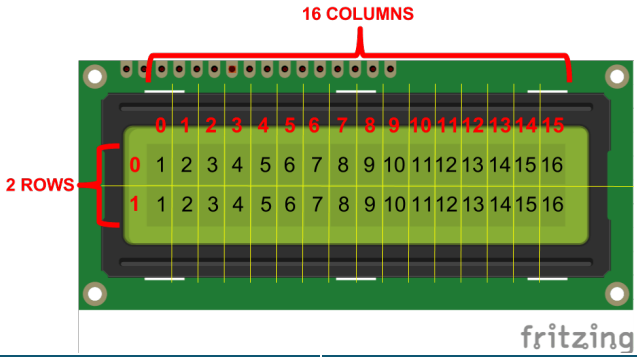


- 체크기는 교실 문 앞에 설치함
- 사람이 지나가는 범위에 초음파센서가 위치하도록 함
- LCD 에는 현재 시각이 표시됨
- 학생은 개인 RFID 카드를 리더기에 태그 한 후 교실에 들어간다
- 카드를 태그 하면 LCD 에는 카드와 매칭되는 학번, 시간, 지각 여부가 표시된다
- 오전 9 시 이후에 태그 하면 지각으로 간주되어 메모리에 학번과 태그 시간이 기록된다
- 초음파센서를 통해 태그 하지 않고 지나가는 경우 (이하 무단침입)를 감지하고 해당 시간이 기록되며 지각자와 구분을 위해 학번 0 번으로 기록된다.
- 정상 통과(지각 안함), 지각, 무단침입 각각의 상황 별로 부저를 통해 멜로디가 재생됨
- 버튼을 통해 저장된 목록을 볼 수 있음.

■ 준비물

부품		센서	
아두이노 우노	RFID 카드	LCD	RTC 모듈 (시계모듈)
브레드보드	USB to 9V DC 케이블	RFID 모듈	부저 (Piezo)
점퍼 케이블	케이스	초음파 센서	조이스틱

■ LCD 출력 포맷

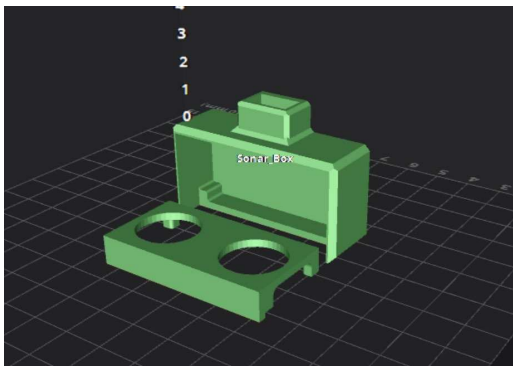


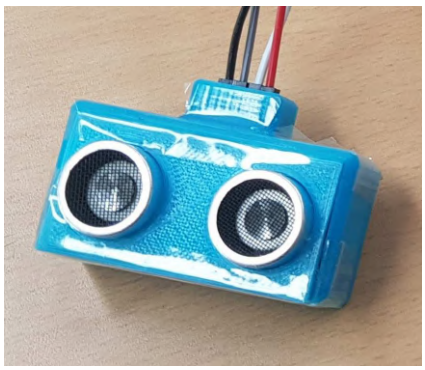
아두이노에 보편적으로 사용되는
16*2 Character LCD 를 사용한다.
1 줄당 16 개의 문자를 총 2 줄에 표시할 수 있으므로,
각각의 상황 별로 다음 표와 같이 출력 포맷을 지정하였다.
지각자 명단 보기의 경우 조이스틱을 이용하여 조작한다.
기본상태에서 조이스틱의 가운데 버튼을 누르면
기록된 지각자 명단이 한 페이지당 2 줄로 나누어 출력된다.
→ 방향으로 움직이면 다음 페이지 표시,
↓ 방향으로 움직이면 명단 보기가 종료, 기본 상태로 돌아온다.

실제 출력	출력 예시	실제 출력	출력 예시
	12.07.2019 11:39:52 **출력 포맷** MM.DD.YYYY HH:MM:SS		TressPass 11:39:52 **출력 포맷** TressPass HH:MM:SS
↑ 기본 상태 (현재시간 표시)		↑ 무단침입 (태그 하지 않고 지나가는 경우)	
실제 출력	출력 예시	실제 출력	출력 예시
	30211 PASS 11:39:52 **출력 포맷** [학번] PASS HH:MM:SS		30207 FAIL 11:39:52 **출력 포맷** [학번] FAIL HH:MM:SS
↑ 정상 통과		↑ 지각했을 때	
실제 출력		출력 예시	출력 포맷
		30207 09:05 30211 09:11	[학번] HH:MM [학번] HH:MM
↑ 기록 목록보기 (지각자 명단)			

■ 초음파 센서 케이스

초음파센서는 위와 같이
개방되어 있다.
센서 보호를 위해 3D 프린터를
이용해 케이스를 제작하였다.





■ 핀 번호 세팅

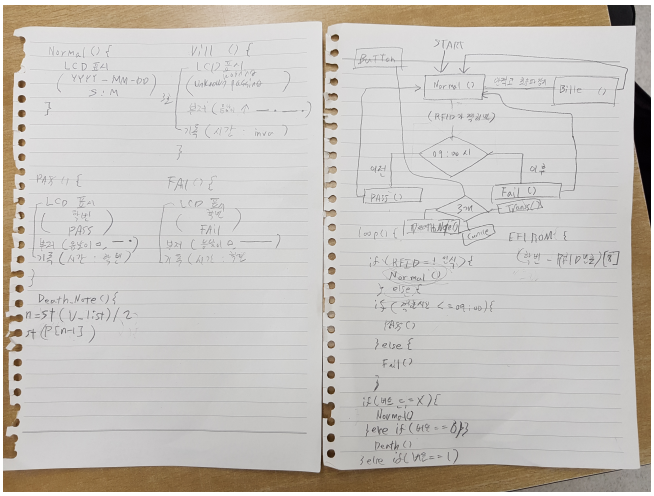
조이스틱			부저	RTC 모듈 (시계 모듈)			초음파 센서		RFID 리더기				
버튼	X 축	Y 축	Piezo	RST	IO	SCK	ECHO	TRIG	RST	SS	SPI		
D2	A0	A1	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13

아두이노에 연결되는 각 센서 별 핀 번호는 다음과 같이 세팅하였다. (D : 디지털 핀, A : 아날로그 핀)

4. 코딩

■ 순서도

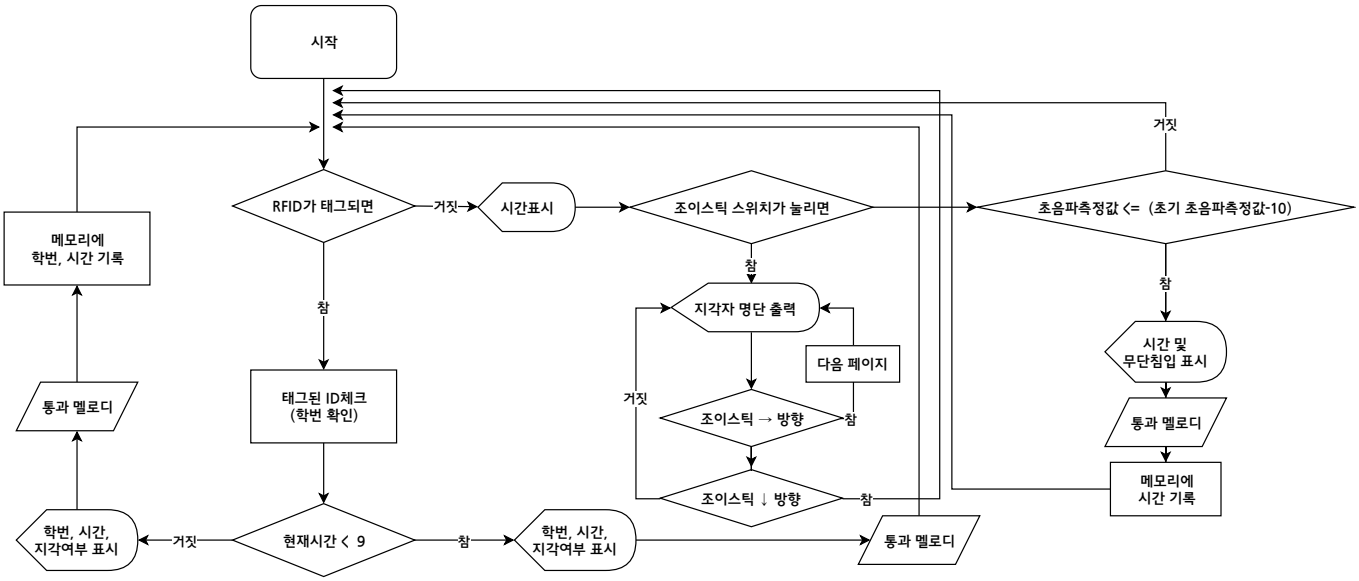
작성자 · 심재빈



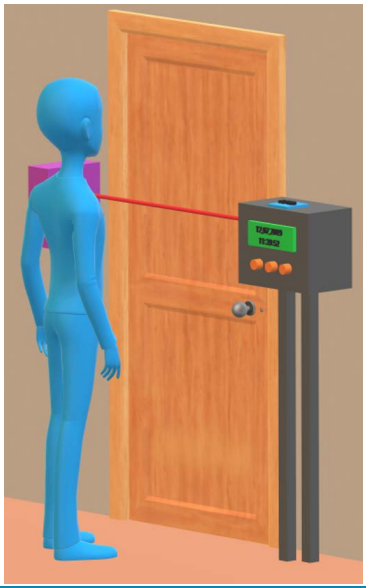
순서도 작성 회의 모습

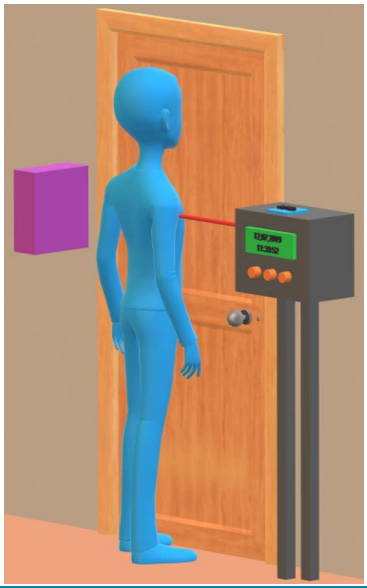
프로그래밍을 하기 전,
논리적인 흐름과 과정 파악을 위해 순서도를 작성하였다.
설계 단계에서 구상한 내용을 표준 기호를 사용해
절차에 맞게 정리함.

왼쪽) 초기 순서도
아래) 최종 순서도



■ 무단침입 체크 알고리즘





평상 시 | 무단침입 시

무단침입 체크를 위해 초음파센서를 이용한다.
초음파센서를 통해 정면에 있는 물체와의 거리를
측정할 수 있는데, (그림의 경우 체크기와 보라색
물체와의 거리) 그림과 같이 문 앞에 사람이
들어오게 되면 초음파센서에서 측정된 거리는
평상 시보다 짧아지게 된다.

◆ 따라서 다음과 같이 알고리즘을 구상하였다.

1. 지각자 체크기에 전원이 들어오면
   곧바로 현재 측정되는 거리 값을 초기값으로
   저장한다. 따라서
   초기값은 문 앞에 사람이 없을 때의 거리가
   된다.
2. 1 초마다 거리 값을 측정하면서
   (초기값-오차범위) >= (현재 거리 값)이
   참이면 무단침입으로 간주한다.
3. 2 번 과정을 무한반복한다.

*오차범위 : 중간에 사람이 아닌 작은 이물질이 개입할 경우를 고려하여 10cm 로 설정함.

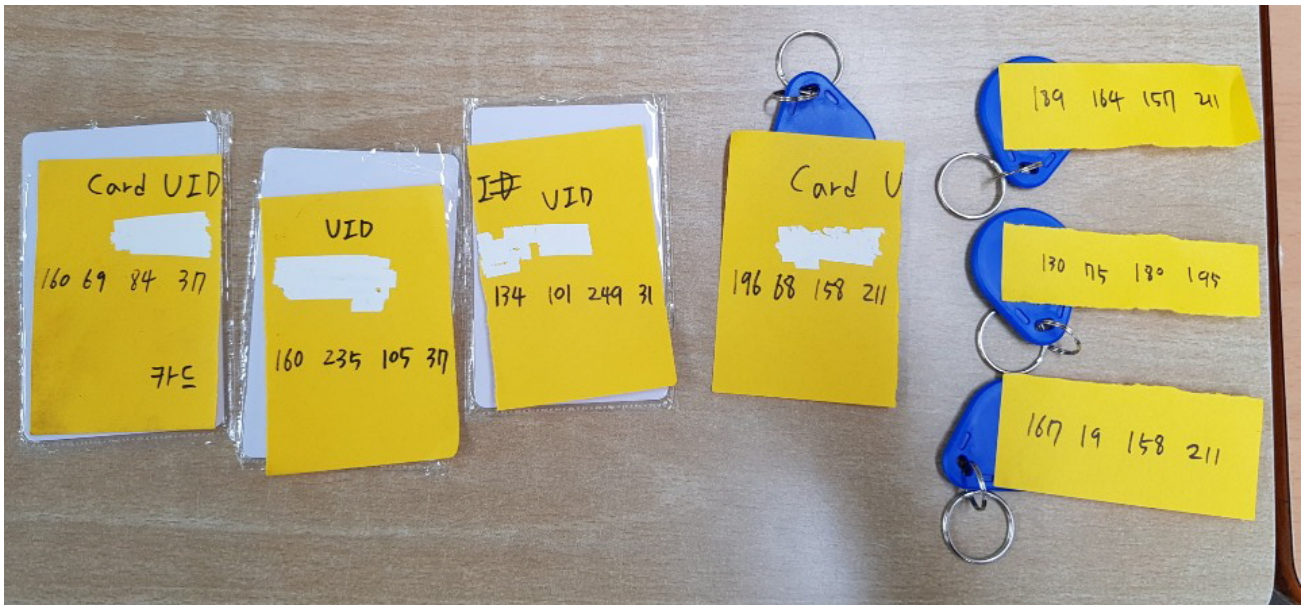
■ 멜로디

RFID 태그 또는 무단침입 시 멜로디를 재생시키기 위해 피에조 부저를 사용하였다.
피에조 부저는 음계를 표현하기위해 수동부저(Passive Buzzer)를 사용하였으며,
수동부저를 음을 내기 위해서는 주파수를 가진 펄스 신호를 부저에게 보내줘야 하기 때문에
다음 표와 같이 각 상황 별 멜로디를 표준 주파수로 바꾸었다.

정상 통과	음계	도#(5), 쉼표, 솔#(5), 도#(5)	박자
	주파수	554, 0, 415, 554	5, 12, 10, 6
지각	음계	파#(5), 파(5), 레#(5), 도#(5), 레#(5), 라#(4), 솔#(4)	박자
	주파수	739, 698, 622, 554, 622, 466, 523, 415	6, 6, 6, 6, 6, 6, 3, 6
무단침입	음계	레(4), 레(4), 레(5), 쉼표, 라(4)	박자
	주파수	293, 293, 587, 0, 440	8, 8, 8, 8, 4

■ RFID 카드

RFID 카드에는 카드 형과 고리 형 두 가지 타입이 있으며, 고유 값인 UID 가 기록되어있다.
UID : 카드 고유 번호로 최대 세 자리 수가 4 개의 묶음으로 이루어져 있다. (XXX XXX XXX XXX)
RFID 모듈을 통해 구입한 RFID 카드의 UID 를 확인하고 다음 표와 같이 학번과 UID 를 임의로 매칭하였다.



학번	UID	학번	UID
30200	무단침입용으로 할당	30204	196 68 158 211
30201	160 69 84 37	30205	167 19 158 211
30202	134 101 249 31	30206	189 164 157 211
30203	160 235 105 37	30207	130 75 180 195

```
int id[7][3] = //id[학번][UID], 세로 인덱스 : 학번, 가로 인덱스 : UID
{0,0,0,160,69,84,37,134,101,249,31,160,235,105,37,196,68,158,211,167,19,158,211,189,164,157,211,1330,75,180,195};
```

세로↓ 가로→	0	1	2	3
0	무단침입용으로 할당			
1	160	69	84	37
2	134	101	249	31
3	160	235	105	37
4	196	68	158	211
5	167	19	158	211
6	180	164	157	211
7	130	75	180	195

위 표를 토대로 id[학번][UID] 포맷으로 2 차원 배열을 선언하였다.

예)
id[1][0] : 1 번 학생의 0 번째 UID : 160
id[3][2] : 3 번 학생의 2 번째 UID : 105
id[7][3] : 7 번 학생의 3 번째 UID : 195

이를 통해 태그 한 RFID 의 UID 4 묶음과 id[n][0] ~ id[n][3]을 비교하여
4 묶음이 모두 일치하는 n(학번)을 찾는 알고리즘을 만들 수 있다.

■ RFID(UID) ↔ 학번 매칭 알고리즘

태그된 RFID 의 UID 4 묶음(XXX XXX XXX XXX)과 id 배열 1~n 학번의 UID 4 묶음(XXX XXX XXX XXX)을 비교하여
4 묶음 모두 일치하는 학번을 찾으면 된다.

```
void id_check() {
  int hop = 0;
  nid = 0;
  for (int i = 1; i < 8; i++) {
    hop = 0;
    for (int j = 0; j < 4; j++) {
      if (mfrc.uid.uidByte[j] == id[i][j]) {
          hop++;
      }
      if (hop == 4) {
          nid = i;
      }
    }
  }
}
```

1. id_check() 함수를 호출했을 때 hop 을 0 으로 초기화
   (예외 처리를 위해 nid 도 0 으로 초기화)
2. 읽어 들인 UID 의 j 번째 묶음과 id[i][j]에서 i 번째 학생의 UID
   j 번째 묶음을 비교한다.
3. 만약 일치하면 hop 을 1 만큼 증가 (hop+1) 시키고
   (hop==4)를 만족하지 않으면 j 를 1 만큼 증가 (j+1)시켜 읽어
   들인 UID 의 j+1 번째 묶음과 i 번째 학생의 UID j+1 번째
   묶음을 비교하는 과정을 반복한다.
4. UID 4 묶음이 모두 일치하면 hop 은 4 번 증가되어
   (hop==4)를 만족하게 된다. (hop==4)를 만족하게 되면
   현재의 i 를 nid(매칭된 학번을 저장하기 위해 선언한 전역
   변수)에 저장하고 함수를 종료한다.
5. 위 과정 중 한 묶음 이라도 일치하지 않으면 hop 을 0 으로
   초기화 (hop=0) 한 뒤 i 를 1 만큼 증가시키고
   2 번 과정으로 돌아가 읽어 들인 UID 의 j 번째 묶음과
   i+1 번째 학생의 UID j 번째 묶음을 비교한다.

따라서 id_check() 함수를 호출하면 전역변수 nid 에는 태그한 학생의 학번이 대입되게 된다.
만약 id 배열과 매칭되지 않는 (등록되지 않은) RFID 를 태그 하였을 경우 위 과정을 거쳐도
nid 에는 1 번 과정의 (nid=0)으로 인해 0 번 학번으로 예외처리 된다.

■ EFPROM 메모리

기록을 위해 아두이노에 내장된 EFPROM(Electrically Erasable Programmable Read-Only Memory)를 이용함.
EFIROM 의 저장 용량은 1KB(1024byte)의 매우 적은 용량이기 때문에 저장할 데이터의 크기를 최소로 줄여야 한다.
저장은 1byte 당 숫자 값은 0~255 까지, 영문자는 알파벳 한 글자만 저장 가능하다.

지각자의 경우 학번(XXXXX)에서 학년, 반(앞 숫자 3 자리)를 제외한 학번(숫자 2 자리)만 기록한다.
태그 시간은 시간(숫자 2 자리) 와 분(숫자 2 자리)를 기록하는 것으로 정하였다.
무단출입의 경우 학번을 0 번으로 간주하고 나머지는 위와 같이 기록한다.

따라서 지각자(또는 무단침입) 한 횟수 당 기록되는 값은 "[학번][시간][분]"으로 3byte 이며
최대 약 341(1024/3) 횟수까지 기록이 가능하게 된다.

■ EFpROM 쓰기

EFPROM 의 기록은 `EFPROM.write(주소, 데이터)` 함수를 이용해 1 차원 배열로 접근하여 기록한다.
저장용량이 1024byte 이므로 주소는 0~1023 까지 있으며 주소와 해당 주소에 저장할 데이터를 인자 값으로 주면 된다.

```
void Write() {
  int n = hop * 3;
  EEPROM.write(n, nid);
  EEPROM.write(n + 1, rtc.getTime().hour);
  EEPROM.write(n + 2, rtc.getTime().min);
  hop++;
}
```

[학번이 30211 인 학생이 09 시 10 분에 태그하여 처음으로 지각한다고 가정하면]
1. 체크기의 전원이 들어오면 전역변수 hop 에 0 을 대입. [(hop=0)]
   *id_check() 함수에 사용된 hop 과는 별개임.
2. id_check() 함수를 호출하여 nid 에 현재 태그 한 학생의 학번이 대입되게 한
   뒤, Write() 함수를 호출함. [(hop=0)]
3. Write() 함수가 호출되면 변수 n 에 hop*3 을 대입. [(hop=0), (n=0)]
4. EEPROM 함수를 이용해 n 번째 주소에 nid 값을 기록한다.
   [(n=0) 이므로 EEPROM 의 0 번째 주소에 nid(학번 11)이 기록됨]
5. n+1 번째 주소에 현재 시간(9)를 기록, n+2 번째 주소에 현재 분(10)을
   기록한다. [(hop=0), (n=0)]
6. hop 을 1 만큼 증가시킨다. [(hop=1), (n=0)]

주소	0	1	2
데이터	11	9	10

위 과정을 거치면 EEPROM 에는 위와 같이 기록된다.

[이어서 학번이 30207 인 학생이 09 시 15 분에 태그 하여 두 번째로 지각한다고 가정하면]

1. 전원이 다시 켜지지 않았으므로 hop 변수는 그대로 둔다. [(hop=1)]
2. Id_check() 함수를 호출한 뒤 Write() 함수를 호출하여 변수 n 에 hop*3 대입. [(hop=1), (n=3), (nid=7)]
3. EEPROM 함수를 이용해 n 번째 주소에 nid 값을 저장한다. [[(hop=1), (n=3), (nid=7)]
4. n+1 번째 주소에 현재 시간(9)를 기록, n+2 번째 주소에 현재 분(15)를 기록한다.
5. Hop 을 1 만큼 증가시킨다. [(hop=2)]

주소	0	1	2	3	4	5
데이터	11	9	10	7	9	15

위 과정을 거치면 EEPROM 에는 위와 같이 기록된다.

이처럼 hop 변수는 기록된 개수(지각자 수 + 무단침입 수)를 뜻하게 하며,
n 변수는 기존에 기록된 데이터를 덮어씌우지 않게 하기 위해, 기록해야 될 주소를 가리키는 포인터 역할을 한다.
무단출입의 경우 0 학번으로 취급하므로 학번만 0 으로 기록되고 나머지는 위 기록 방식과 같다.

■ EFPROM 읽기

주소 인자를 넣어주면 해당 주소의 데이터를 반환하는 `EFPROM.read(주소)` 함수를 이용함.
LCD 출력 포맷에서 구상한 대로, 기록된 데이터를 한 페이지당 2 줄로 페이지를 나누어 표시한다.
조이스틱을 → 방향으로 움직이면 다음 페이지를 표시하고, ↓ 방향으로 움직이면 기록 보기를 종료하고
기본 상태로 돌아오게 만든다.

Print_Page() 함수

코드는 13 페이지의 Print_Page() 함수 참조.

표시할 줄(l)과 EEPROM 에서 읽기 시작할 주소(j)를 인자로 받아,
기록된 데이터를 한 페이지당 2 줄로 나누어 동적으로 표시하는 Print_Page() 함수를 만들었다.

사용 예)
Print_Page(0, 0) : 첫 번째 줄에 EEPROM 0 번째 주소부터 2 번째 주소까지 (학번, 시간, 분)를 동적으로 표시.
Print_Page(1, 3) : 두 번째 줄에 EEPROM 3 번째 주소부터 5 번째 주소까지 (학번, 시간, 분)를 동적으로 표시.
따라서 이 함수는 l 의 범위가 0~1 이며, j 는 3 의 배수인 인자 값만 받을 수 있다.

EEPROM 쓰기과정에서 학번은 5 자리(예 30211)중 302 를 제외한 번호(11)만 기록하였었음으로,
출력 시에는 번호 앞에 302 를 붙여서 출력한다. 기록된 학번이 한 자리인 경우(예 5 번)는 "05"와 같이 출력되도록
(학번<10)이 참이면 앞에 0 을 붙여서 출력, 학번이 두 자리인 경우 그대로 출력되도록 하였다. 이는 시간과 분 출력 시에도 마찬가지로 적용하였다.

List() 함수

코드는 13 페이지의 List() 함수 참조.

호출하면 현재 EEPROM 에 기록된 개수에 따라 Print_Page()에 적절한 인자 값을 넘겨 호출하는 List() 함수를 만들었다.

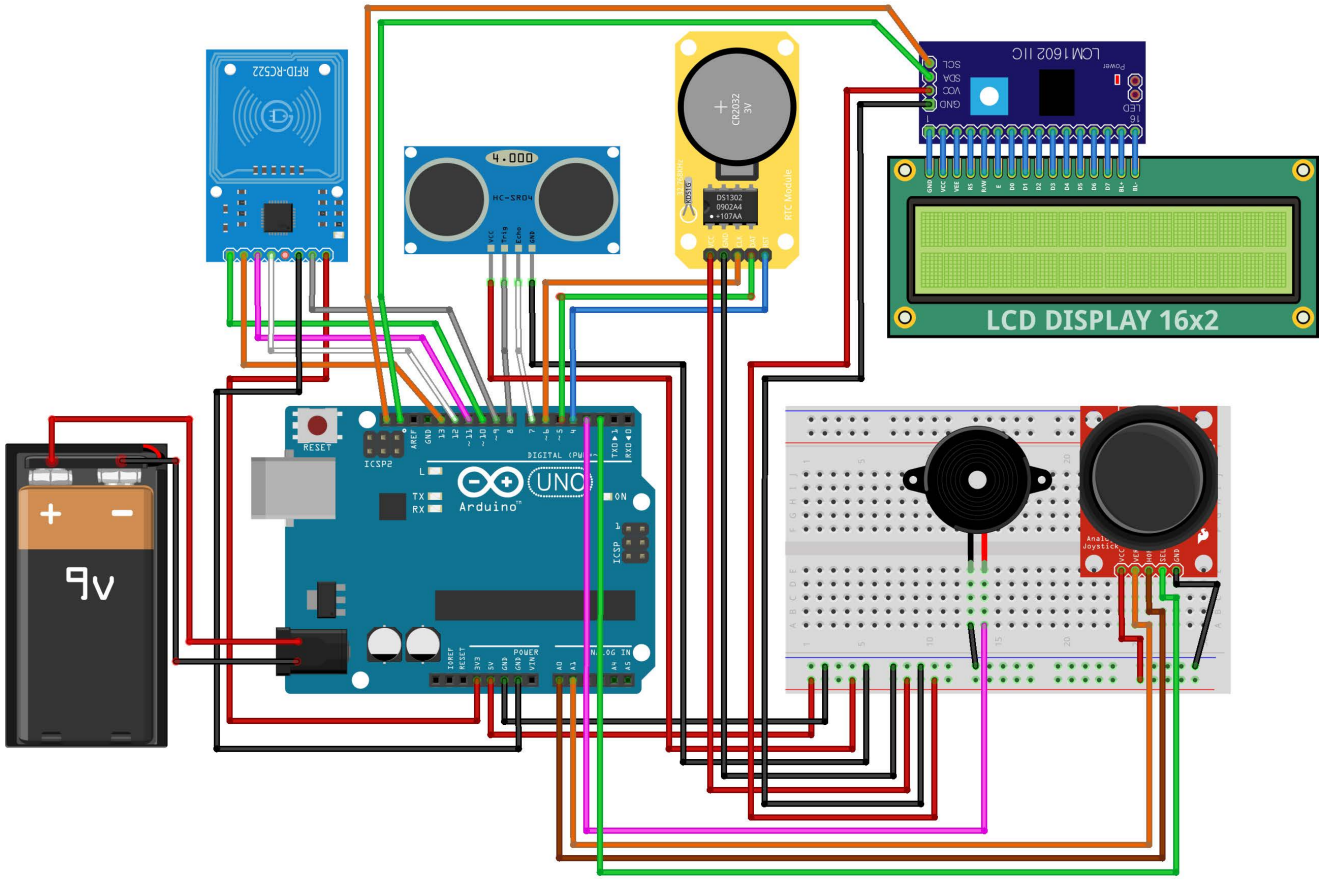
1. 조이스틱 가운데 버튼이 눌리면 List() 함수가 호출된다.
2. 위에서 언급했듯이 전역 변수 hop 은 기록된 개수를 뜻하므로, switch()문을 통해 hop 값의 따라 갈리도록 하였다.
   hop=0 : 기록된 데이터가 없으므로 "NO DATA" 출력.
   hop=1 : 1 개만 표시하면 되므로 페이지를 나눌 필요가 없다. 따라서 한 명만 출력되도록 Print_Page(0, 0) 호출.
   hop=2 : 2 개만 표시하면 되므로 페이지를 나눌 필요가 없다. 따라서 한 명만 출력되도록 Print_Page(0, 0), Print_Page(1, 3) 호출.
   그 외 (hop>=3) : 페이지를 나눠야 하므로 과정 3 진행.
3. i 를 0 으로 초기화 하고, j 에 i*6 대입. [(i=0), (j=0)].
4. Print_Page(0, j) - 첫 번째 줄에, EEPROM i*6 번째 주소부터 시작되는 데이터 출력. [(j=0)이므로 0 번째 주소부터 시작되는 데이터]
5. Print_Page(1, j+3) - 두 번째 줄에, EEPROM (i*6)+3 번째 주소부터 시작되는 데이터 출력.
   [(j=0)이므로 3 번째 주소부터 시작되는 데이터]
6. 조이스틱을 → 방향으로 움직이면 7 번 과정, ↓ 방향으로 움직이면 함수를 종료하고 기본상태로 돌아옴.
7. i 를 1 만큼 증가시키고, j 에 i*6 대입 [(i=1). (j=6)].
8. Print_Page(0, j) - 첫 번째 줄에, EEPROM i*6 번째 주소부터 시작되는 데이터 출력. [(j=6)이므로 6 번째 주소부터 시작되는 데이터]
9. Print_Page(1, j+3) - 두 번째 줄에, EEPROM (i*6)+3 번째 주소부터 시작되는 데이터 출력.
   [(j=6)이므로 9 번째 주소부터 시작되는 데이터]
10. 조이스틱을 → 방향으로 움직이면 7 번 과정 [(i=2),(j=12)]가 됨, ↓ 방향으로 움직이면 함수를 종료하고 기본상태로 돌아옴.

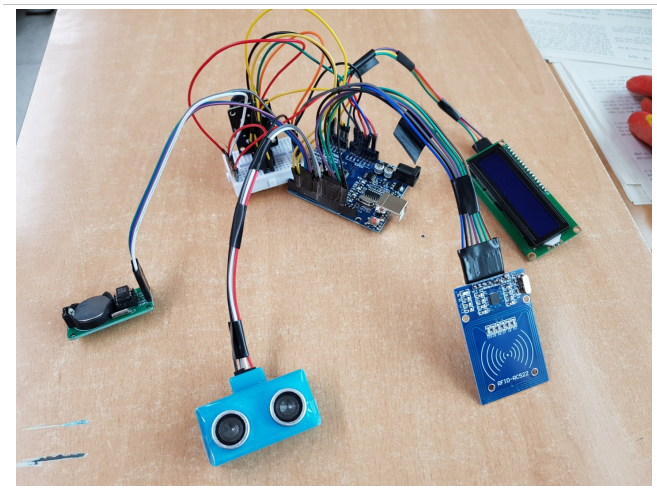
위 과정을 통해 페이지를 넘길 때 마다 Print_Page() 함수의 j 값을 3 의 배수로 증가시키며 대입해 기록된 데이터가 순차적으로 표시되게 된다. 7~9 번 과정을 반복하면서 (i<=hop)이 만족할 때마다 i 를 다시 0 으로 초기화 시켜,
기록된 주소 이상으로 접근하지 않도록 예외처리를 하였다.

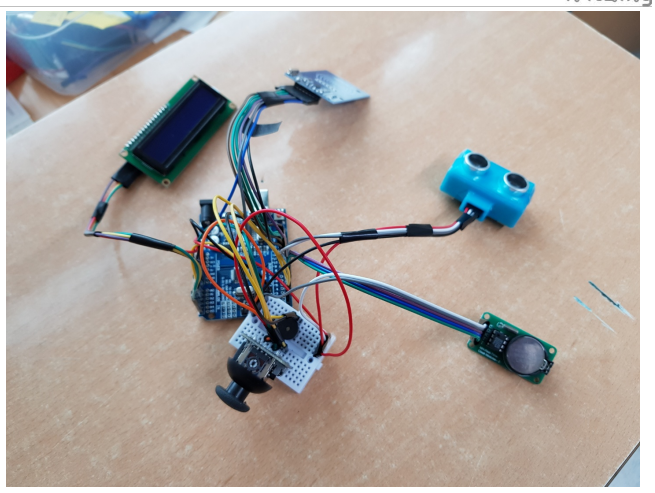
5. 제작

■ 회로도

작성자 · 심재빈





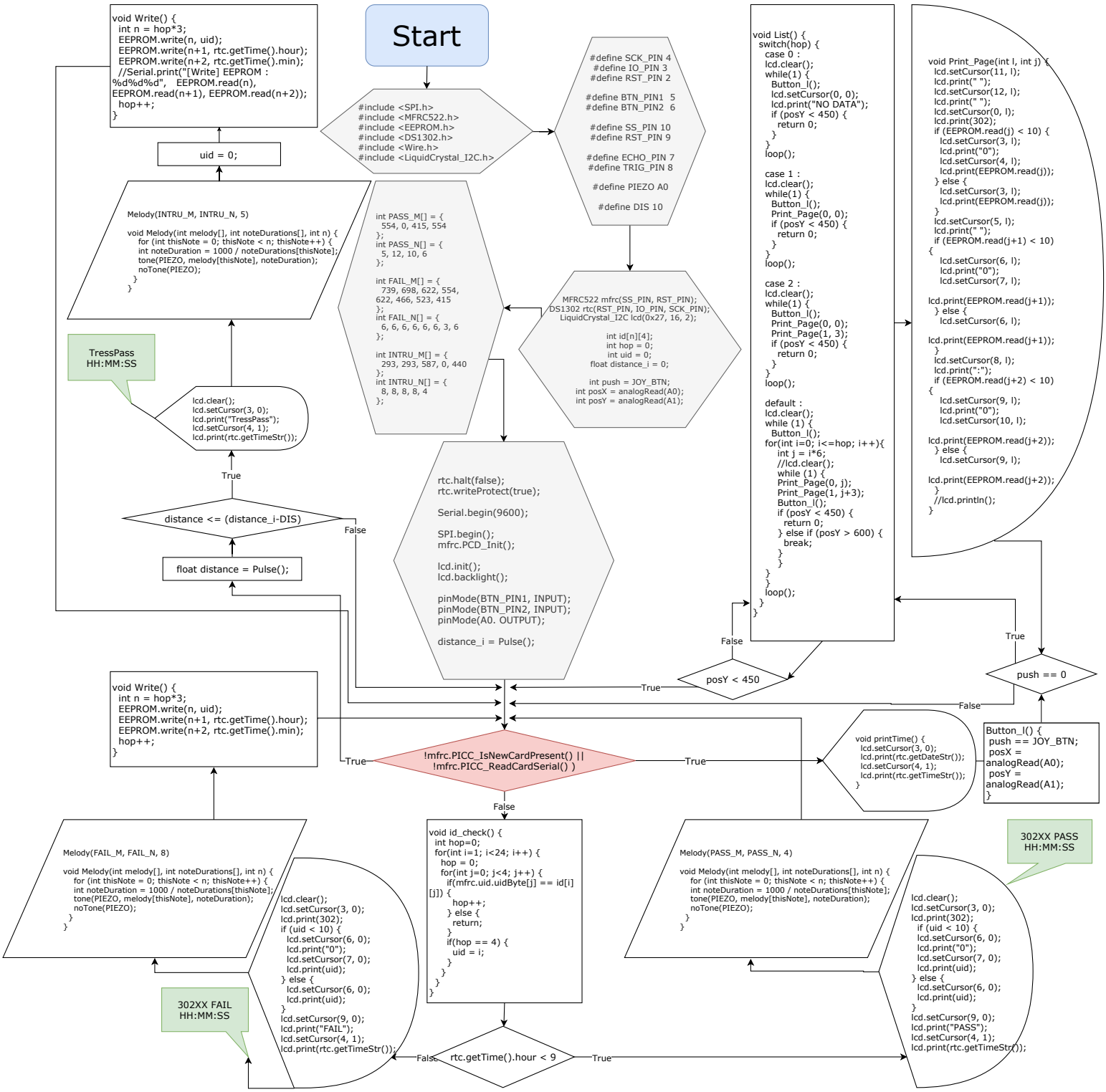


회로도는 위와 같다. (맨 위 왼쪽부터 RFID 모듈, 초음파 센서, RTC 모듈, LCD)
아래는 회로도를 토대로 실제 제작한 지각자 체크기 회로 모습사진이다. (좌, 우 다른 각도)
전력공급은 9V 전지를 이용하거나 USB 5v to DC 9v 케이블을 이용해 보조배터리로 공급한다.

■ 소스코드 순서도

작성자 · 심재빈

소스코드를 기반으로 표준 기호에 맞게 다음과 같은 순서도를 그림.



```c
#include <SPI.h>
#include <MFRC522.h>
#include <EEPROM.h>

#include <DS1302.h> // DS1302 RTC 라이브러리
#include <Wire.h>  // Wire 라이브러리
#include <LiquidCrystal_I2C.h> // LCD 라이브러리

//핀 세팅
#define JOY_BTN 2 //조이스틱 버튼
#define PIEZO 3 //피에조

//RTC 모듈
#define RST_PIN 4
#define IO_PIN 5
#define SCK_PIN 6

//초음파센서
#define ECHO_PIN 7
#define TRIG_PIN 8

//RFID 모듈
#define RST_PIN 9
#define SS_PIN 10

//초음파센서 오차범위 (단위 CM)
#define DIS 10

MFRC522 mfrc(SS_PIN, RST_PIN);
//DS1302 rtc(RST_PIN, IO_PIN, SCK_PIN);
DS1302 rtc(4, 5, 6);
LiquidCrystal_I2C lcd(0x27, 16, 2);

//학번배열 선언 (학번이 1~8번인 학생 8명인 경우, 0번은 무단침입으로 간주)
int id[8][4] = {0,0,0,0,160,69,84,37,134,101,249,31,160,235,105,37,196,68,158,211,167,19,158,211,189,164,157,211,130,75,180,195};
int hop = 0; //학번매칭에 사용될 홉
int nid = 0; //태그된 UID와 매칭되는 학번 저장용
float distance_i = 0; //초음파센서 초기값

//조이스틱 초기값
int push = JOY_BTN;
int posX = analogRead(A0);
int posY = analogRead(A1);

//상황별 멜로디 음계
/* PASS : 정상통과 (지각자 아닐경우)
 *  FAIL : 지각자
 *  INTRU : 무단침입 (태그안하고 그냥 지나감)
 */
int PASS_M[] = {
  554, 0, 415, 554
};
int PASS_N[] = {
  5, 12, 10, 6
};

int FAIL_M[] = {
  739, 698, 622, 554, 622, 466, 523, 415
};
int FAIL_N[] = {
  6, 6, 6, 6, 6, 6, 3, 6
};

int INTRU_M[] = {
  293, 293, 587, 0, 440
};
int INTRU_N[] = {
  8, 8, 8, 8, 4
};

void setup(){
  rtc.halt(false);
  rtc.writeProtect(true);

  Serial.begin(9600);
  SPI.begin();
  mfrc.PCD_Init();

  lcd.init();
  lcd.backlight(); //LCD ON

  //초음파 센서
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  //조이스틱
  pinMode(2, INPUT_PULLUP);
  //피에조
  pinMode(3, OUTPUT);
  //초음파센서 초기값 저장
  distance_i = Pulse();
  /*
  rtc.setDOW(THURSDAY); // 요일 설정
  rtc.setTime(15, 59, 50); // 시간 설정, 순서대로 (시간, 분, 초)
  rtc.setDate(16, 6, 2019); // 날짜 설정, 순서대로 (일, 월, 년)
```

```
*/
}

void loop(){
  //Button_l();
  if ( !mfrc.PICC_IsNewCardPresent() || !mfrc.PICC_ReadCardSerial() ) {
    // RFID 태그가 되지 않았을때 또는 ID가 안 읽혀질 때
    Normal(); //기본상태 (LCD에 현재시간 출력)
    Button(); //버튼입력 감지 (조이스틱)
    INTRU_check(); //무단침입 체크
    Serial.println("loop if end");
  } else { //RFID 태그되었을때
    id_check(); //RFID UID와 매칭되는 학번(id) 찾기
    printUID(); //시리얼로 태그된 UID 확인용
    if ( rtc.getTime().hour < 9 ) { //태그 한 시각이 9시 이전이면
    PASS(); //정상통과 처리(지각자 아님)
    } else { // 9시 이후면
    FAIL(); //지각자 처리
    }
  }
}

void Button_l(){ //현재 입력되는 버튼(조이스틱)값 확인
push = JOY_BTN;
posX = analogRead(A0);
posY = analogRead(A1);
}

void Write() { //EEPROM에 지각자 기록
  int n = hop*3; //기록할 때 [학번,시간,분] 3바이트씩 기록하므로
  EEPROM.write(n, nid); //학번 저장
  EEPROM.write(n+1, rtc.getTime().hour); //시간 저장
  EEPROM.write(n+2, rtc.getTime().min); //분 저장
  hop++;
}

void id_check() { //태그된 RFID UID와 매칭되는 학번 찾기
  int hop=0;
  nid = 0; //현재 태그된 학번(nid)에 기본값 0으로 초기화
  for(int i=1; i<8; i++) { //학생이 8명이므로
    hop = 0;
    for(int j=0; j<4; j++) { //UID가 숫자 3자리 3묶음이므로
      if(mfrc.uid.uidByte[j] == id[i][j]) {
        hop++;
      }
      if(hop == 4) { //hop이 4가 되면 태그된 UID 4묶음이 모두 선언된 학번 매칭값과 일치
        nid = i; //현재 태그된 학번(nid)에 학번(i) 저장
      }
    }
  }
}

void INTRU_check(){ //무단침입 체크
  Serial.println("INTRU_check()");
  float distance = Pulse(); //현재 측정되는 초음파센서 값 저장
  if(distance <= (distance_i-DIS)) { //현재 측정된 초음파센서값을 초기값(초기 작동시 측정값 - 오차범위)와 비교
    INTRU(); //무단침입 간주
  }
  Serial.println("INTRU NO");
}

void printTime() { //LCD와 시리얼에 현재시간 표시
  lcd.setCursor(3, 0);
  lcd.print(rtc.getDateStr());
  lcd.setCursor(4, 1);
  lcd.print(rtc.getTimeStr());

  Serial.print("날짜 : ");
  Serial.print(rtc.getDOWStr());
  Serial.print(" ");
  Serial.print(rtc.getDateStr());
  Serial.print(" ");
  Serial.print("시간: ");
  Serial.println(rtc.getTimeStr());
  Serial.print("hop : ");
  Serial.println(hop);
  delay(1000);  //1초
}

void printUID() { //태그된 RFID UID, NID값 출력
  Serial.print("Card UID : ");
  for (byte i = 0; i < 4; i++) {
    Serial.print(mfrc.uid.uidByte[i]);
    Serial.print(" ");
  }
  Serial.println();
  Serial.print("NID : ");
  Serial.println(nid);
}

void Button() { //현재 입력된 버튼(조이스틱)값 확인
  Button_l(); //현재 버튼 입력값으로 초기화
  if (push == 0) { //조이스틱 버튼이 눌리면
  List(); //저장된 명단보기
}
```

```
}

float Pulse(){ //초음파센서 값 계산 (단위 CM)
  digitalWrite(TRIG_PIN, LOW);
  digitalWrite(ECHO_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  unsigned long duration = pulseIn(ECHO_PIN, HIGH);
  float distance = ((float)(340 * duration) / 10000) / 2;
  return distance;
}

void Print_Page(int l, int j) { //저장된 목록을 LCD 최대 출력범위 2줄 당 1페이지로 분할하여 LCD에 출력
  //l은 표시할 줄번호, j는 EEPROM에서 읽기 시작할 배열주소
  //LCD 출력 전 이미 출력된 문자를 없애는 과정
  lcd.setCursor(11, l);
  lcd.print(" ");
  lcd.setCursor(12, l);
  lcd.print(" ");
  lcd.setCursor(0, l); //LCD의 0번째 행, l번째 줄에
  lcd.print(302); //EEPROM에는 학번 5자리 중 끝 2자리 번호만 저장되므로 3-2반 학생 공통인 302를 출력
  if (EEPROM.read(j) < 10) { //학번이 한 자릿수이면 십의 자리에 0을 붙여서 출력
    lcd.setCursor(3, l);
    lcd.print("0");
    lcd.setCursor(4, l);
    lcd.print(EEPROM.read(j));
  } else { //학번이 두 자릿수면 그대로 출력
    lcd.setCursor(3, l);
    lcd.print(EEPROM.read(j));
  }
  lcd.setCursor(5, l);
  lcd.print(" ");
  if (EEPROM.read(j+1) < 10) { //시간이 한 자릿수이면 십의 자리에 0을 붙여서 출력
    lcd.setCursor(6, l);
    lcd.print("0");
    lcd.setCursor(7, l);
    lcd.print(EEPROM.read(j+1));
  } else { //시간이 두 자릿수면 그대로 출력
    lcd.setCursor(6, l);
    lcd.print(EEPROM.read(j+1));
  }
  lcd.setCursor(8, l);
  lcd.print(":");
  if (EEPROM.read(j+2) < 10) { //분이 한 자릿수이면 십의 자리에 0을 붙여서 출력
    lcd.setCursor(9, l);
    lcd.print("0");
    lcd.setCursor(10, l);
    lcd.print(EEPROM.read(j+2));
  } else { //분이 두 자릿수면 그대로 출력
    lcd.setCursor(9, l);
    lcd.print(EEPROM.read(j+2));
  }
}

void List() { //기록된 명단출력
  switch(hop) { //hop : EEPROM에 기록된 학생 수
    case 0 : //기록된게 없으면 (0명)
    lcd.clear();
    while(1) { //LCD에 출력유지
      Button_l();
      lcd.setCursor(0, 0);
      lcd.print("NO DATA");
      if (posY < 450) { //조이스틱을 ←으로 움직이면 출력종료하고 기본상태로 돌아감
        return 0;
      }
    }
    loop();

    case 1 : //1명 기록되있으면
    lcd.clear();
    while(1) { //LCD에 출력유지
      Button_l();
      Print_Page(0, 0); //첫 번째 줄에, EEPROM 배열 0부터 시작되는 데이터 출력
      if (posY < 450) { //조이스틱을 ←으로 움직이면 출력종료하고 기본상태로 돌아감
        return 0;
      }
    }
    loop();

    case 2 : //2명 기록되있으면
    lcd.clear();
    while(1) { //LCD에 출력유지
      Button_l();
      Print_Page(0, 0); //첫 번째 줄에, EEPROM 배열 0부터 시작되는 데이터 출력
      Print_Page(1, 3); //두 번째 줄에, EEPROM 배열 3부터 시작되는 데이터 출력
      if (posY < 450) { //조이스틱을 ←으로 움직이면 출력종료하고 기본상태로 돌아감
        return 0;
      }
    }
    loop();

    default : //3명 이상이면 (LCD에 2명까지만 출력가능하므로 페이지를 나눠야 함)
    lcd.clear();
```

```
    while (1) { //LCD에 출력유지
      Button_l();
    for(int i=0; i<=hop; i++){ //EEPROM에 기록된 학생 수(hop)만큼 반복
        int j = i*6;
        while (1) { //LCD에 출력유지
        Print_Page(0, j); //첫 번째 줄에, EEPROM의 i*6번째 배열부터 시작되는 데이터 출력
        Print_Page(1, j+3); //두 번째 줄에, EEPROM의 (i*6)+3번째 배열부터 시작되는 데이터 출력
        Button_l();
        if (posY > 600) { //조이스틱을 →으로 움직이면 다음 페이지 출력 //posY < 450
          return 0;
        } else if (posX < 450) { //조이스틱을 ↓으로 움직이면 출력종료하고 기본상태로 돌아감
          break;
        }
        }
    }
    }
    loop();
  }
}

void Melody(int melody[], int noteDurations[], int n) { //피에조 멜로디 출력(음계, 박자, 출력 음 개수)
    for (int thisNote = 0; thisNote < n; thisNote++) {
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(PIEZO, melody[thisNote], noteDuration);
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(PIEZO);
  }
}

void Normal() { //기본상태
  Serial.println("Normal()");
  lcd.clear();
  printTime(); //현재시간 출력
}

void PASS() { //정상통과 처리(지각안함)
  lcd.clear();
  lcd.setCursor(3, 0);
  lcd.print(302); //302출력
    if (nid < 10) { //학번이 한 자릿수이면 십의 자리에 0을 붙여서 출력
    lcd.setCursor(6, 0);
    lcd.print("0");
    lcd.setCursor(7, 0);
    lcd.print(nid);
  } else { //학번이 두 자릿수면 그대로 출력
    lcd.setCursor(6, 0);
    lcd.print(nid);
  }
  lcd.setCursor(9, 0);
  lcd.print("PASS"); //LCD에 PASS 출력
  lcd.setCursor(4, 1);
  lcd.print(rtc.getTimeStr());
  Melody(PASS_M, PASS_N, 4); //피에조 멜로디 출력
  delay(3000);
}

void FAIL() { //지각자 처리
  lcd.clear();
  lcd.setCursor(3, 0);
  lcd.print(302); //302출력
  if (nid < 10) { //학번이 한 자릿수이면 십의 자리에 0을 붙여서 출력
    lcd.setCursor(6, 0);
    lcd.print("0");
    lcd.setCursor(7, 0);
    lcd.print(nid);
  } else { //학번이 두 자릿수면 그대로 출력
    lcd.setCursor(6, 0);
    lcd.print(nid);
  }
  lcd.setCursor(9, 0);
  lcd.print("FAIL"); //LCD에 FAIL 출력
  lcd.setCursor(4, 1);
  lcd.print(rtc.getTimeStr());
  Melody(FAIL_M, FAIL_N, 8); //피에조 멜로디 출력
  Write(); //EEPROM에 저장
  delay(1000);
}

void INTRU() { //무단침입 처리
  lcd.clear();
  lcd.setCursor(3, 0);
  lcd.print("TressPass"); //LCD에 TressPass 출력
  lcd.setCursor(4, 1);
  lcd.print(rtc.getTimeStr());
  nid = 0; //현재 학번을 0으로 간주
  Write(); //학번 0으로 EEPROM에 저장
  Melody(INTRU_M, INTRU_N, 5); //피에조 멜로디 출력
  delay(1000);
}
```