사 업 계 획 서

1 팀원의 현황

1-2 대표 및 팀원의 구성 (팀원이 더 많으면 추가하세요. 최대 4명)

대표	성 명	김소림	연락처	핸드폰	010-7106-7263
	학 번	1615008		이메일	ksr10217@naver.com
	소 속	이화여자대학교 컴퓨터공학과			
	역 할	processing, server, DB			
팀원1	성 명	이효정	연락처	핸드폰	010-2346-1511
	학 번	1771047		이메일	lhj97033@naver.com
	소 속	이화여자대학교 컴퓨터공학과			
	역 할	미끄럼틀, 그네 arduino			
팀원2	성 명	현재정	연락처	핸드폰	010-4158-2528
	학 번	1615079		이메일	wowjd123589@naver.com
	소 속	이화여자대학교 컴퓨터공학과			
	역 할	JSON데이터 전송, 미끄럼틀과 그네 제작, 문서/영상작업			

2 창업동기 및 창업계획

2-1. 창업동기 및 신청사유

저희들은 더 나은 세상을 만들어가기 위해 모였습니다. 컴퓨터공학 기술로 많은 사람들에게 도움이 된다면 더 나은 세상을 만들어가는데에 도움이 될 것입니다. 이를 위해 사람들이 일상에서 겪는 문제점이 무엇인지 파악하고자 노력하였습니다.

실내놀이터인 키즈카페 성행과 함께 낙후되어가는 옥외놀이터를 보면서 옥외놀이터만의 강점이 가려지는 것에 안타까움을 느꼈습니다. 현재 각광받고 있는 실내놀이터의 안전성을 옥외놀이터에서도 구현하고자 세 명이 팀을 이루어 약 1년동안 고민하였습니다.

옥외놀이터에서 단 한명의 아이라도 다치면 안된다는 신념으로 제품개발에 임하고 있습니다. 한 가정의 아이가 즐겁게 놀다가 불의의 사고를 당하지 않도록 최선을 다하겠습니다.

2-2. 신청과제에 대한 신청자의 전문성 기재

컴퓨터공학과 4학년 학생들이 모여 신청과제를 진행하였습니다. 강서구청 스마트도시 아이디어 공모전에서 장려상을 수상하였습니다. 양천구청 스마트도시 공모전에서도 장령상을 수상하였습니다. 또한 관악구 스마트 시티 공모전의 결과를 기다리는 중입니다.

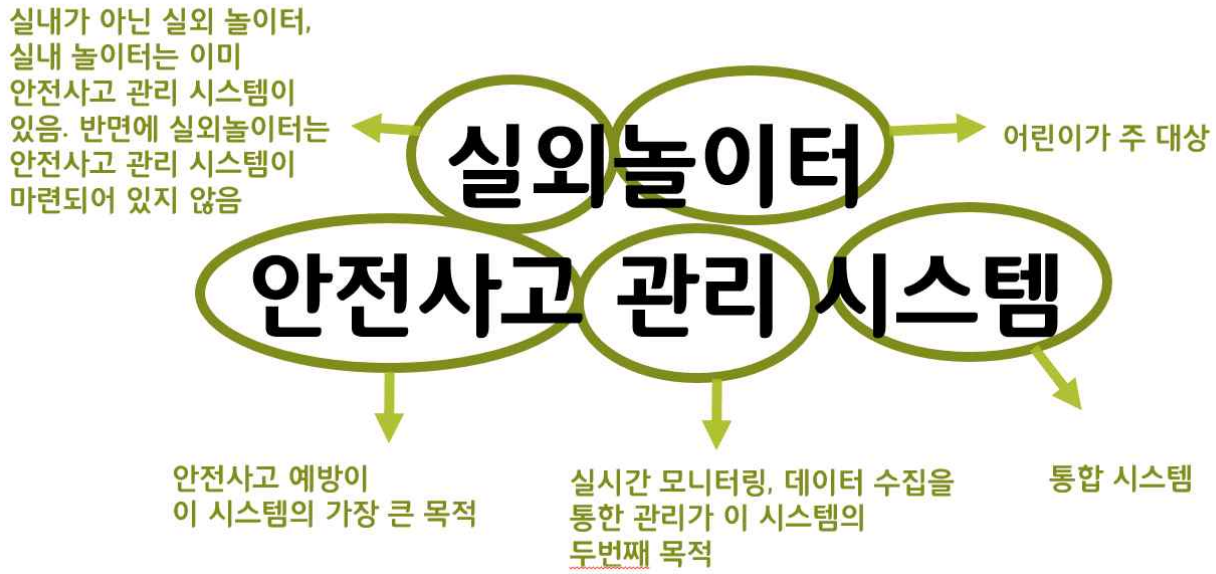
2-3. 활동 계획

현재 여러 창업 경진대회, 공모전에 출전하여 창업을 위한 자금을 모으고 있습니다. 현재 3명의 개발자로 팀을 이루고 있기 때문에 각자 맡은 분야의 전문성을 키우기 위해 공부를 하고 있습니다. 각자의 분야에서 전문성을 키운 후 졸업하면, 3명이 팀을 이루어 정부사업을 신청할 예정입니다. 다양한 경로로 자금을 마련하고 사업을 확장시키기 위하여 뜻이 맞는 홍보마케팅, 경영 담당 등을 영입할 예정입니다. 어느 정도의 수익이 나기 시작하면 한국엔젤투자협회의 투자를 받아 사업을 진행할 예정입니다.

3	사업계획서 요약

3-1. 우리의 서비스 또는 사업을 한마디로 정의

IoT기반 옥외놀이터 모니터링 및 안전관리 시스템



3-2. 고객이 가지고 있는 문제점을 정의

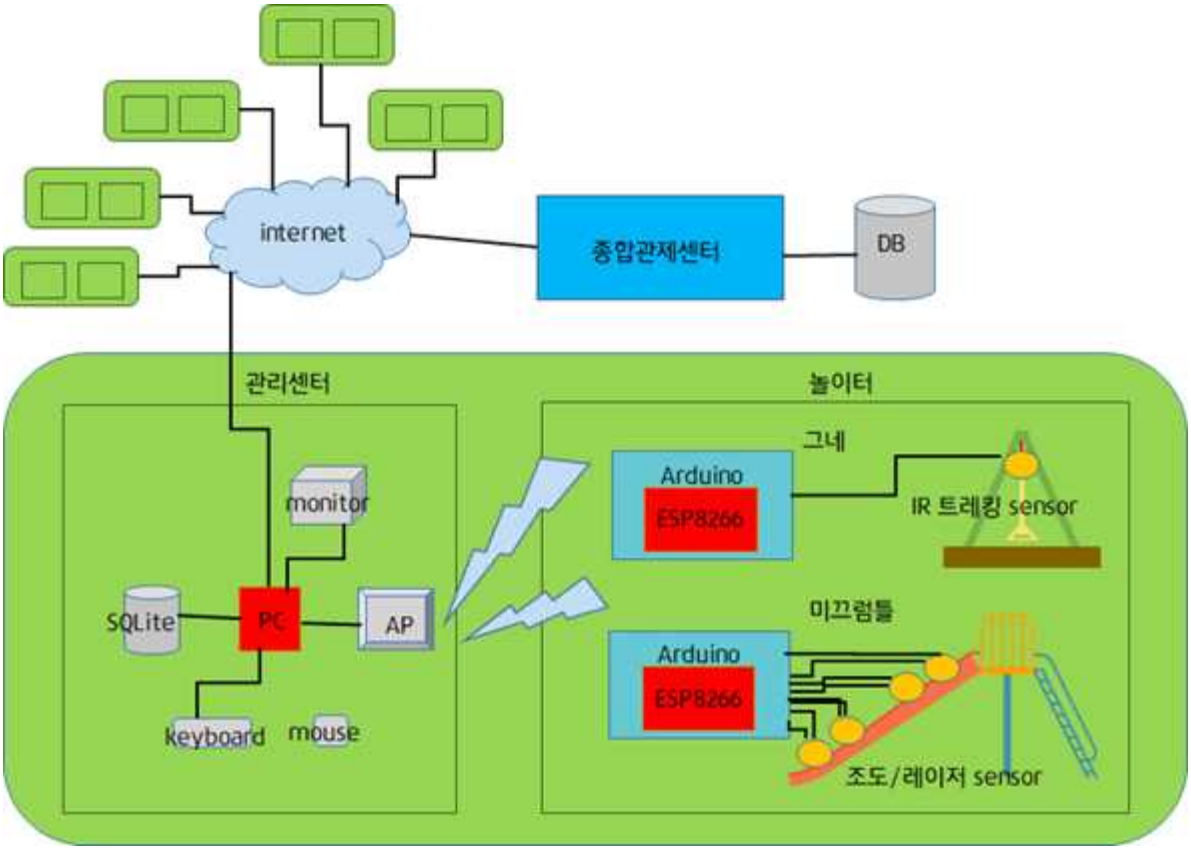
어린이 놀이시설 안전사고는 집 앞에 설치된 놀이터에서 가장 많이 발생하고 있다. 행정안전부는 2018년 기준으로 놀이시설 내 안전사고가 286건 발생한다고 밝혔다. 사고사례를 분석한 결과 안전사고 286건 중에서 237명 정도가 골절 등의 부상을 입었다. 사고원인을 살펴보면, 이용자 부주의로 인한 사고가 282건(98.7%)으로 가장 많았다는 것을 알 수 있다. 이에 따라 우리는 위험 상황을 나누어 이용자에게 주의를 줄 수 있는 시스템을 고안해 내었다.

위험 상황으로 선정한 주요 사고 내용의 구체적 예시들은 다음과 같다. 미끄럼틀의 사고 경우는 미끄럼틀을 거꾸로 오르다 내려오던 아이와 충돌 및 추락하여 골절상하는 경우와 도착지점에 있는 아이와 충돌 후 뒤로 넘어져 다리 인대 파열된 사고가 있었다. 그네의 경우는 10세 여아가 그네 옆을 지나가다 그네를 높이 타던 아이와 충돌해 골절상을 입은 사고가 있었다.

이처럼 놀이터에서의 부주의로 일어난 사고가 제일 많기 때문에 미리 알람을 주어 경각심을 일으키는 것이 중요하다. 사고율을 줄이고 부상을 최소화하기 위하고자 이처럼 문제를 정의하였다.

3-3. 해결방안 - 우리만의 방법, 차별성

시스템 구성도

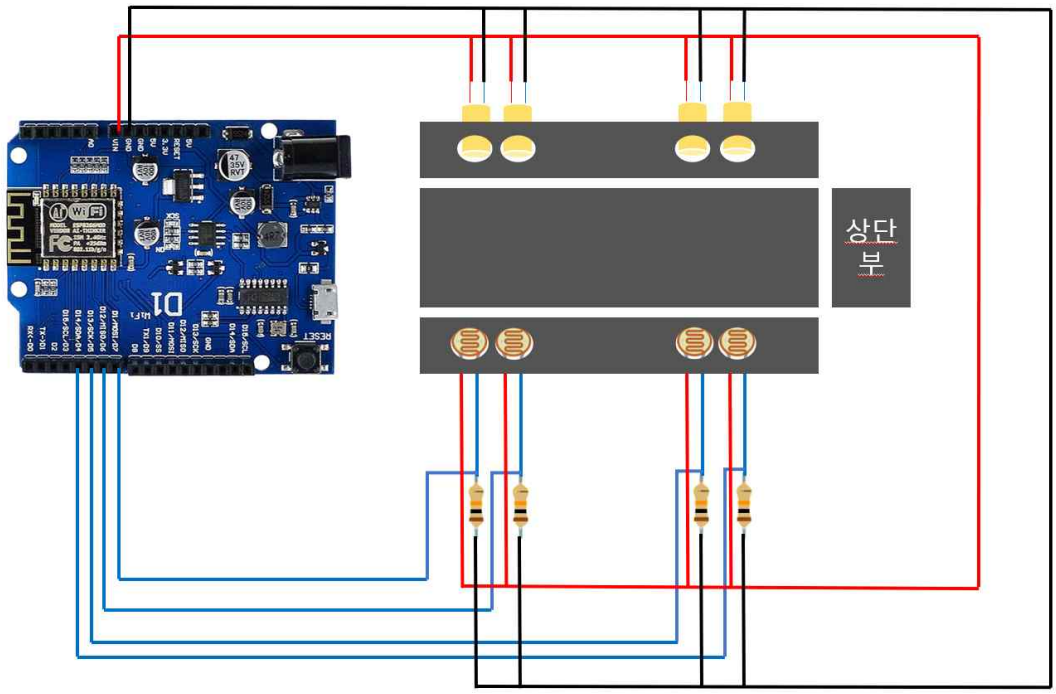


<시스템 구성도>

시스템 구성도이다. 놀이터에 있는 기구 중 가장 사고가 빈번히 일어나는 미끄럼틀과 그네에 집중하였다. 각 놀이터에는 그네와 미끄럼틀에 esp8266 D1보드가 부착되어 있다. 그네는 IR트래킹 센서를 이용하였고 미끄럼틀은 조도 센서와 레이저 센서를 사용하였다. 수집한 정보는 공유기를 통해 관리자 화면으로 전달되고 데이터베이스에 저장된다. 이 각각의 놀이터에 있는 시스템들은 인터넷을 통해 최종적으로 종합관제센터에 모여 저장되는 구조를 가지고 있다.

아두이노 구현방법

미끄럼틀에서는 레이저 센서와 조도센서를 미끄럼틀 상단부, 하단부 옆면에 각각 2개씩 평행하게 설치한다.(미끄럼틀 오른쪽 옆면에는 레이저센서를, 왼쪽 옆면에는 조도센서를 설치한다.) 미끄럼틀 바닥으로부터 3cm 높이에 설치하며, 2개의 레이저 센서 간격은 2cm이다. 센서는 미끄럼틀 옆면 안에 설치되어 사용자는 센서를 볼 수 없다. 햇빛 등의 영향을 받지 않기 위해 조도센서와 레이저 센서에서 1cm 앞까지 빛을 반사하지 않는 검은 관을 설치한다. 레이저 센서는 조도센서로 빛을 내보낸다. 조도센서는 레이저 센서의 빛을 받아 1의 값을 출력한다. 사람이 지나가면 레이저 빛이 가려 0의 값을 출력한다. 센서는 다음 그림과 같이 위에서부터 A, B, C, D번 센서로 명칭한다.



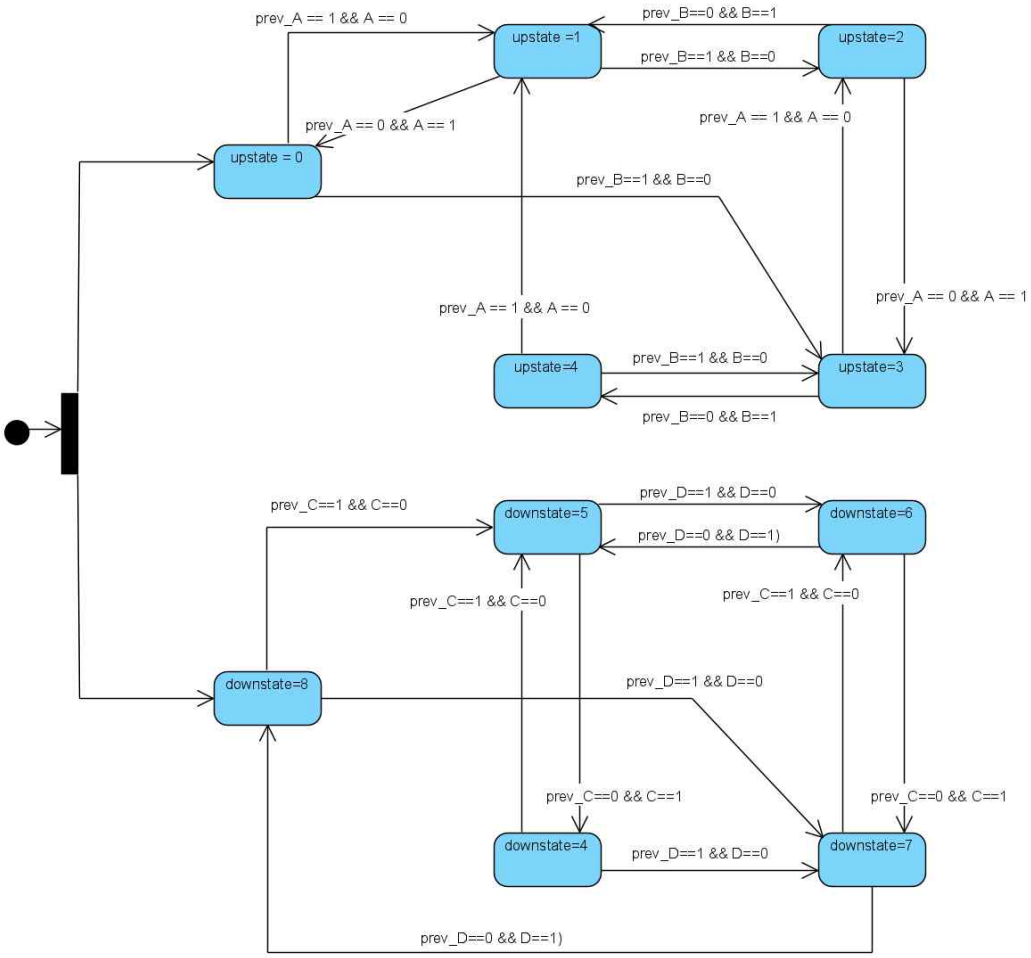
<미끄럼틀 그림-상단부가 미끄럼틀 윗면이고 센서는 미끄럼틀 옆면에 설치되어있다>

센서는 Esp8266 D1보드와 연결되어있다. Esp8266 D1보드는 센서의 값을 읽고, 사용자가 어떤 상태로 미끄럼틀을 타고 있는 지 파악하여 데이터를 processing으로 보낸다. Esp8266 D1보드 역시 미끄럼틀 안쪽에 설치하여 사용자는 볼 수 없다.

Esp8266 D1보드는 센서의 값을 이용하여 사람이 미끄럼틀 어느 위치에 있는지 판단하고, 사람이 어떤 위치에서 위치로 옮겨갔는지의 정보로 사람이 미끄럼틀을 타는 경우를 판단한다.

센서를 기준으로 사람이 미끄럼틀에서 있을 수 있는 위치는 다음과 같이 9가지다.

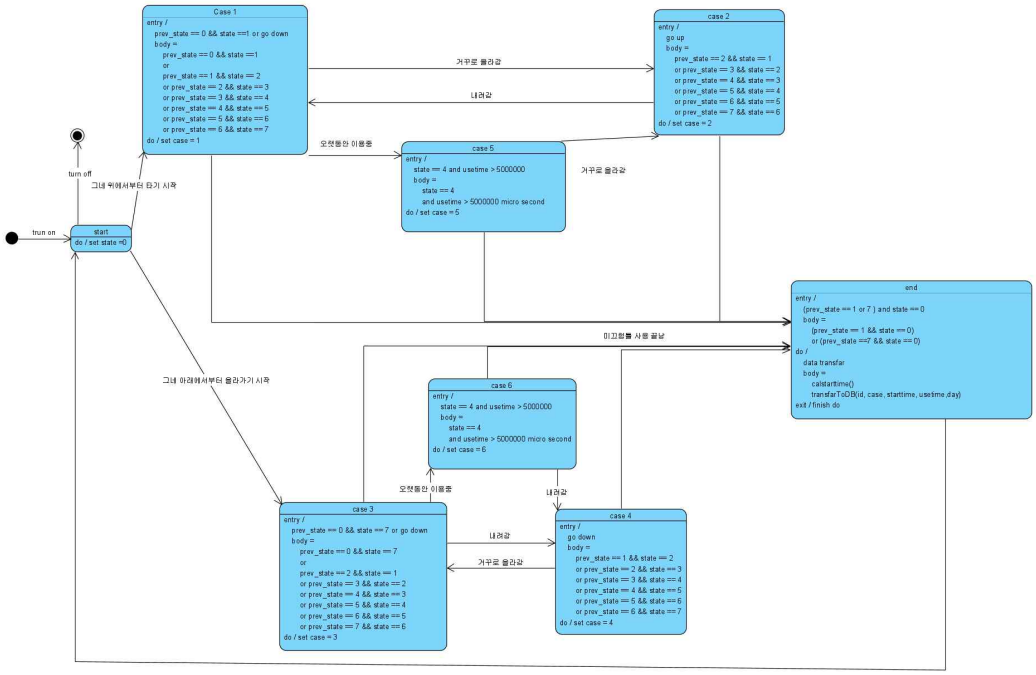
0. 미끄럼틀 이용 전(후) 미끄럼틀 상단부에 있음
1. A센서를 가리고 있음
2. A,B 센서를 가리고 있음
3. B센서를 가리고 있음
4. B,C센서 사이에서 어떠한 센서도 가리고 있지 않음
5. C센서를 가리고 있음
6. C,D센서를 가리고 있음
7. D센서를 가리고 있음
8. 미끄럼틀 이용 후(전) 미끄럼틀 하단부에 있음



위와 같이 statemachine으로 표현한다. 사람이 움직이면 현재 위치에서 변할 수 있는 가지수는 몇 개 안되기 때문에 이를 이요하여 state를 정한다. 미끄럼틀 상단부와 하단부에서 따로 state를 결정한다. 예시를 들어 statemachine을 설명하자면, state결정을 시작하면 upstate와 downstate 두 개에 대해 state를 도출한다. upstate는 기본적으로 0으로 세팅하고 downstate는 8로 세팅한다. 사람이 A 센서로 진입을 시도하면 A의 센서값이 1에서 0이 되므로 upstate를 1로 바꾼다. 그 다음 사람이 계속 내려가면 B센서값도 1에서 0으로 바뀌므로 upstate는 2가 된다. 그 다음 사람이 더 내려가서 A 센서로부터 벗어나면 A센서값은 0에서 1로 바뀌고 upstate는 3이 된다. 사람이 미끄럼틀 상단부의 센서를 모두 지나면 B센서값도 0이 되어 upstate는 4가 된다.

사람이 미끄럼틀을 타는 경우는 아래와 같이 6가지로 정한다.

1. 정상적으로 내려감
2. 내려가다가 올라감
3. 밑에서부터 올라감
4. 밑에서부터 올라가다가 내려감
5. 내려가다가 중간에서 오랫동안 멈춰있다 내려감
6. 올라가다가 중간에서 오랫동안 멈춰있다 올라감



위와같이 statemachine으로 표현한다.(state를 정할때에는 0과 8을 구분했지만 실제로 0과 8은 같은 상태, 즉 미끄럼틀을 이용하지 않는 상태이므로 모두 0으로 표시한다.) 예를 들어 statemachine을 표현하자면, 아두이노가 작동하면 case는 start(state 0)상태로 세팅한다. 사람이 미끄럼틀 상단부에서 타고 내려오면 state가 1이 되고, case1의 진입조건을 만족하여 case를 1로 세팅한다. 그렇게 미끄럼틀을 계속 타다가 미끄럼틀 중반부(state 4)에서 오래동안 움직이지 않으면 사용시간이 길어지고 이 시간이 일정시간보다 길명 case 5의 진입조건을 만족하여 case는 5가 된다. 이대로미끄럼틀을 계속 타고 내려가면 state가 7에서 0이 되고 미끄럼틀 이용이 끝난다. 미끄럼틀 사용에 대한 데이터를 서버로 전송하고 start로 돌아간다.

<주요 코드 서술>

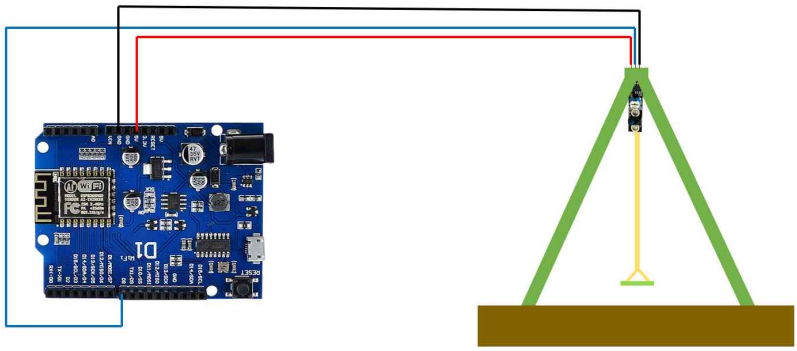
void loop() : statemachine에서 설명한 state와 case를 결정하는 함수

void sendmessage() : 미끄럼틀 이용이 끝나면 'id/case/시작시각/요일/사용시간'을 서버로 보내는 함수

void sendstatemessage() : 실시간 미끄럼틀 센서값을 서버로 전송하는 함수

void caltime() / void settime() : google로부터 시간을 받아 대한민국 시간으로 바꾸고, 필요할 경우 아두이노 자체시간을 더해 시작시각을 계산한다.

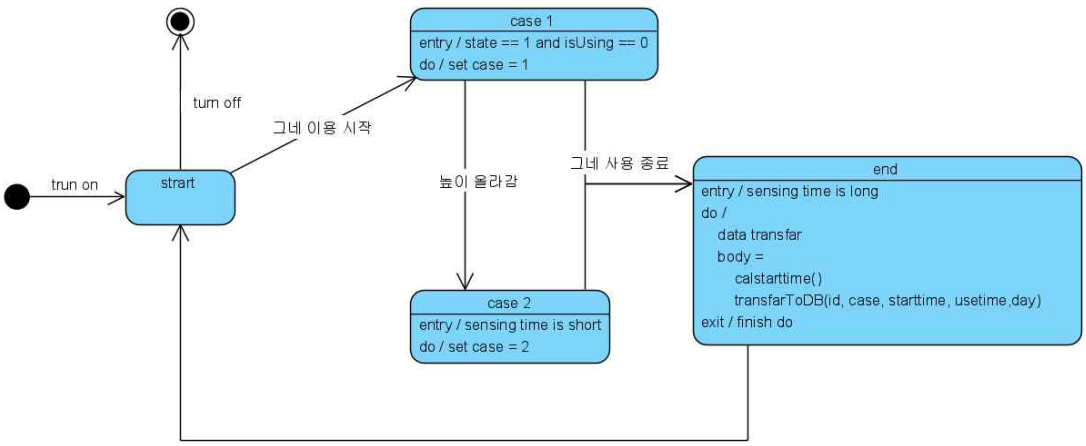
그네에서는 IR트래킹 센서를 사용한다. 센서는 그네 줄의 시작부분의 가로지지대로부터 10cm 부분에 가로지지대와 연직 방향으로 설치한다. 멈춰있는 그네 줄과 평행하며, 그네줄과의 간격은 1cm이다. IR트래킹 센서는 적외선을 내보내고 반사된 적외선의 양을 인식한다. 그네줄에 반사된 적외선을 읽으면 0값을 출력하고 그네줄이 없으면 반사된 물체가 없어 1값을 출력한다. 적외선 반사가 잘 되도록 그네줄에 반사스티커를 부착한다.(IR센서가 보이는 방향으로 부착한다.)



<그네 그림-IR트래킹센서가 잘보이게 센서를 반대로 보여준다. 실제로는 센서의 보이는 부분이 그네줄을 향해야 한다>

센서는 Esp8266 D1보드와 연결되어있다. Esp8266 D1보드는 센서의 값을 읽고, 사용자가 어떤 상태로 그네를 타고 있는지 파악하여 데이터를 processing으로 보낸다. Esp8266 D1보드는 그네 가로지지대에 설치하여 사용자의 손이 닿지 않게 한다.
Esp8266 D1보드는 센서의 값을 이용하여 사람이 그네를 어떻게 이용중인지 판단하고, 해당 정보를 이용하여 사람이 그네를 타는 경우를 판단한다.

사람이 그네을 타는 경우는 아래와 같이 2가지로 정한다.
1. 정상적으로 그네를 탐
2. 너무 높이 올라감



위와같이 statemachine으로 표현한다. 예시를 들어 설명하면, 아두이노가 시작하면 start가 된다. 그네줄이 인식된 상태(state 0)로 세팅되어있고 그네줄이 움직여 state가 1이 되고 그네가 사용중이 아니면 그네 사용을 시작하고 case가 1이 된다. 그네를 타다가 높이 올라간 상황이면 그네줄이 센서를 지나는 시간(그네줄의 두께만큼 인식되는 시간)이 일정시간보다 짧으면 높이올라간 것이라 판단하여 case를 2로 세팅한다. 줄이 멈추고 그네줄이 인식되는 시간이 일정시간보다 길면 그네 이용이 끝난 것이라 판단하고 서버로 데이터를 보낸다.

Esp8266 D1보드는 2가지의 데이터를 모아 processing으로 전송한다. 첫 번째는 실시간 이용 정보, 즉 센서의 원초값이다. 센서로부터 값을 읽고 이 값들을 '/'로 구분하여 UDP 패킷으로 processing 서버에 보낸다. 그네의 경우 센서가 하나지만 '센서값/높이 올라간 여부'을 보낸다. 두 번째는 미끄럼틀(그네) 사용이 끝난 사용자가 미끄럼틀(그네)을 어떤 경우로 이용했는지 정보다. 해당 정보를 모아 '/'로 구분하여 UDP 패킷으로 processing 서버에 보낸다. 정보는 '미끄럼틀(그네)ID/이용한 경우의 번호/미끄럼틀(그네) 이용 시작시각/요일/이용시간/'이다.

이용시간은 Esp8266 D1보드 자체 시간을 사용하고 시간 단위는 microsecond이다.

미끄럼틀(그네) 이용 시작시각은 다음과 같이 정한다. 1시간마다 goolge에서 시간을 받아와 한국시간으로 변경한다. 단위는 '년-월-일-시-분-초(1/1000000초 까지 표현)'이다. 이후 시작시각이 필요할때 Esp8266 D1보드 자체시간(microsecond단위)을 읽어 기존에 계산된 시각에 더하여 '년-월-일-시-분-초'에 맞게 계산한다.

<코드 서술>

void buttonLoop() : statemachine에 따라 그네가 어떤 상태로 이용중인지 판단하는 함수

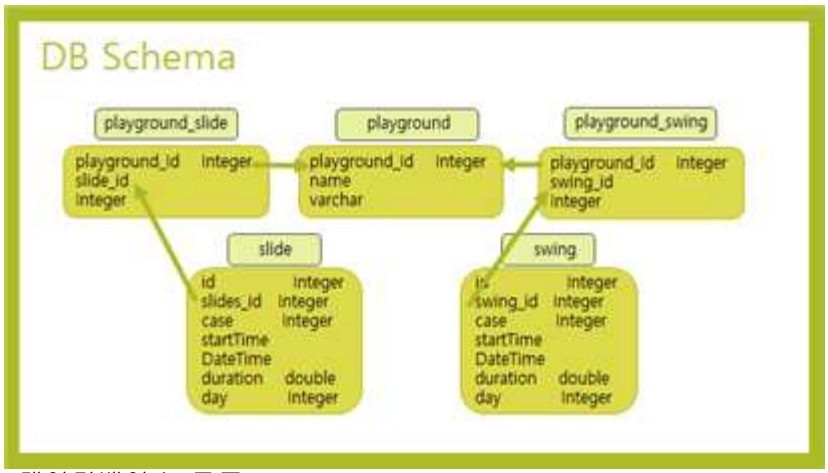
void sendstatemessage() : 그네의 실시간 정보를 보낸다. '센서값/그네가 높이 올라간여부'로 데이터를 모아 보낸다.

void sendmessage() / void caltime() / void settime() : 미끄럼틀과 코드는 같다.

Database

이 프로젝트는 SQLite DB를 사용하고 있다. SQLite를 사용한 이유는 다음과 같다. 프로젝트의 초기 버전이고 놀이터 관리자들만 사용하기 때문에 사용하는 user 수에 제한이 없다. 사용자 수가 크게 문제 되지 않는 상황에서 mySQL과 SQLite를 비교했을 때, SQLite가 더 가볍고 속도가 빠르다. 두 번째로 SQLite는 embedded 된 형태로 존재하는 것이 많아 다루기 수월하다. 우리가 서버로 사용할 processing에 SQLite 기능이 포함되어 있었다. 세 번째는 SQLite는 무료로 사용할 수 있어 학생인 우리에게 부담이 없다.

이 프로젝트는 총 6개의 table, day_time, my_configuration, playground_slide, playground_swing, slides, swing table을 가지고 있다.



<데이터베이스 구조>

swing table은 id, swing_id, case, startTime, duration, day 가 있는 테이블로 id가 primary key이면서 auto increment 값이다. 이 table은 arduino에서 판단되어 전송된 케이스와 startTime, duration, 요일 등을 저장하여 후에 web page를 통해 통계자료를 보여줄 때 쓰일 테이블이다.

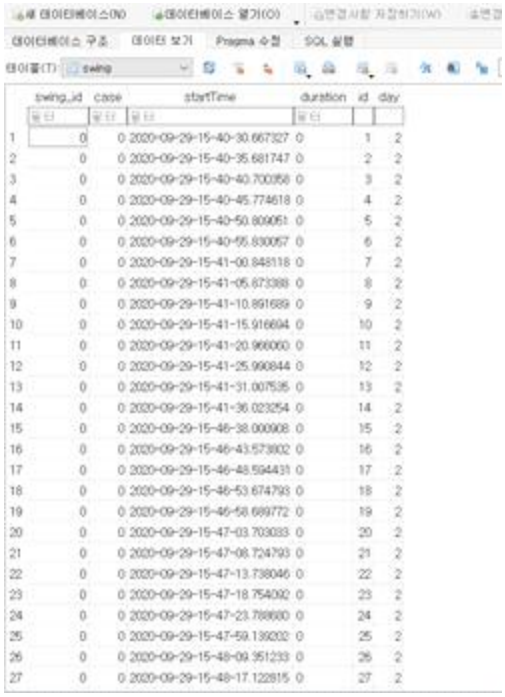
slides table은 id, slides_id, case, startTime, duration, day가 있는 테이블로 id가 primary key이면서 auto increment 값이다. 이 table은 arduino에서 판단되어 전송된 케이스와 startTime, duration, 요일 등을 저장하여 후에 web page를 통해 통계자료를 보여줄 때 쓰일 테이블이다.

playground_slide table은 playground_id와 slide_id가 있는 table로 primary key는 playground_id와 slide_id이다. 각 놀이터 내에 어떤 미끄럼틀이 있는지 알려주는 테이블이다.

playground_swing table은 playground_id와 swing_id가 있는 table로 primary key는 playground_id와 swing_id이다. 각 놀이터 내에 어떤 그네가 있는지 알려주는 테이블이다.

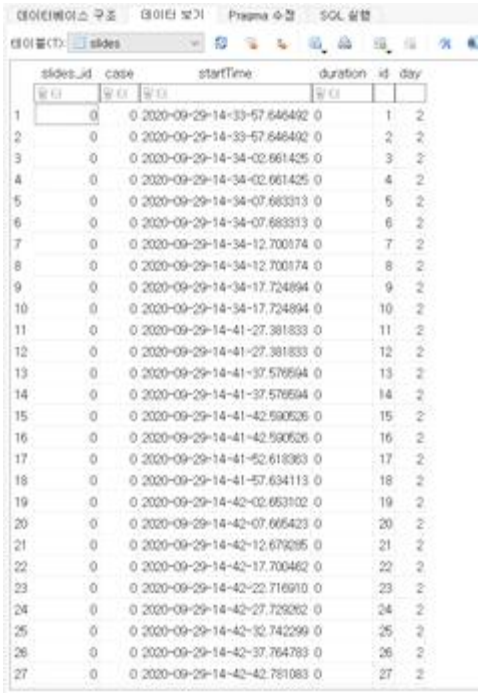
day_time table은 day, time, value 값이 있는 table이다. 이 테이블은 백업을 위한 테이블이라고 볼 수 있는데, 이 테이블은 slides table, swing table에서 한번 가공된 정보를 저장하고 있다. 프로젝트의 web page에서 보여주는 정보가 누적 정보이기 때문에 이전 정보를 모두 가지고 있어야 하는데 이전 정보를 swing table이나 slides table에서 모두 불러와 다 계산하기에는 계산량이 많을 수 있다. 이 테이블은 미리 계산한 정보를 저장해두어 계산량을 줄이는 역할을 한다.

my_configuration table은 opening date와 opening_day가 있는 table이다. 시작날짜와 시작요일을 담고 있는 테이블이다. 이 테이블에 있는 시작날짜와 시작요일을 기준으로 날짜와 요일을 계산하게 된다.



	swing_id	case	startTime	duration	id	day
1	0	0	2020-09-29-15-40-30.667327	0	1	2
2	0	0	2020-09-29-15-40-35.681747	0	2	2
3	0	0	2020-09-29-15-40-40.700358	0	3	2
4	0	0	2020-09-29-15-40-45.774618	0	4	2
5	0	0	2020-09-29-15-40-50.809051	0	5	2
6	0	0	2020-09-29-15-40-55.830057	0	6	2
7	0	0	2020-09-29-15-41-00.848118	0	7	2
8	0	0	2020-09-29-15-41-05.873388	0	8	2
9	0	0	2020-09-29-15-41-10.891689	0	9	2
10	0	0	2020-09-29-15-41-15.916694	0	10	2
11	0	0	2020-09-29-15-41-20.966060	0	11	2
12	0	0	2020-09-29-15-41-25.990844	0	12	2
13	0	0	2020-09-29-15-41-31.007535	0	13	2
14	0	0	2020-09-29-15-41-36.023254	0	14	2
15	0	0	2020-09-29-15-46-38.000908	0	15	2
16	0	0	2020-09-29-15-46-43.573802	0	16	2
17	0	0	2020-09-29-15-46-48.594431	0	17	2
18	0	0	2020-09-29-15-46-53.674793	0	18	2
19	0	0	2020-09-29-15-46-58.689772	0	19	2
20	0	0	2020-09-29-15-47-03.703033	0	20	2
21	0	0	2020-09-29-15-47-08.724793	0	21	2
22	0	0	2020-09-29-15-47-13.738046	0	22	2
23	0	0	2020-09-29-15-47-18.754092	0	23	2
24	0	0	2020-09-29-15-47-23.768680	0	24	2
25	0	0	2020-09-29-15-47-59.139202	0	25	2
26	0	0	2020-09-29-15-48-09.951233	0	26	2
27	0	0	2020-09-29-15-48-17.122815	0	27	2

<ex) swing table>



	slides_id	case	startTime	duration	id	day
1	0	0	2020-09-29-14-33-57.646492	0	1	2
2	0	0	2020-09-29-14-33-57.646492	0	2	2
3	0	0	2020-09-29-14-34-02.661425	0	3	2
4	0	0	2020-09-29-14-34-02.661425	0	4	2
5	0	0	2020-09-29-14-34-07.683313	0	5	2
6	0	0	2020-09-29-14-34-07.683313	0	6	2
7	0	0	2020-09-29-14-34-12.700174	0	7	2
8	0	0	2020-09-29-14-34-12.700174	0	8	2
9	0	0	2020-09-29-14-34-17.724894	0	9	2
10	0	0	2020-09-29-14-34-17.724894	0	10	2
11	0	0	2020-09-29-14-41-27.381833	0	11	2
12	0	0	2020-09-29-14-41-27.381833	0	12	2
13	0	0	2020-09-29-14-41-37.576594	0	13	2
14	0	0	2020-09-29-14-41-37.576594	0	14	2
15	0	0	2020-09-29-14-41-42.590526	0	15	2
16	0	0	2020-09-29-14-41-42.590526	0	16	2
17	0	0	2020-09-29-14-41-52.613363	0	17	2
18	0	0	2020-09-29-14-41-57.634113	0	18	2
19	0	0	2020-09-29-14-42-02.653102	0	19	2
20	0	0	2020-09-29-14-42-07.665423	0	20	2
21	0	0	2020-09-29-14-42-12.679285	0	21	2
22	0	0	2020-09-29-14-42-17.700462	0	22	2
23	0	0	2020-09-29-14-42-22.716910	0	23	2
24	0	0	2020-09-29-14-42-27.729262	0	24	2
25	0	0	2020-09-29-14-42-32.742299	0	25	2
26	0	0	2020-09-29-14-42-37.764783	0	26	2
27	0	0	2020-09-29-14-42-42.781083	0	27	2

<ex) slides table>

Server

다음은 DB와 연동되는 서버를 구성하는 코드이다.

void setup() 함수 : 어떤 포트에서 서버를 열지, 어떤 프로토콜을 통해 데이터를 받을 것인지 정해준다. udp1 = new UDP(this, 8100); udp2 = new UDP(this, 8200); s1 = new Server(this, 8100); s2 = new Server(this, 8200);

서버가 동작하고 DB와 연결이 되면 DB에 있는 정보를 가공하는 작업을 한다. web page에 표시하고 싶은 내용은 2가지이다. 각 케이스가 몇 번 나타나는지, 요일, 시간대별로 그네/미끄럼틀 사용량이 얼마나 되는지이다. 각 케이스가 몇 번 나타나는지 알기 위해 다음 쿼리를 사용한다.

db.query("SELECT ₩"case₩", count(id) as cnt FROM slides group by ₩"case₩"");

case별로 알기 위해 group by를 사용하였다.

요일, 시간대별로 그네/미끄럼틀 사용량이 얼마나 되는지 알기 위해 다음 쿼리를 사용한다.

db.query("SELECT count(id) as cnt FROM slides where startTime like ₩""+ddate+"%₩""); 이 쿼리는 for문을 통해 0시부터 24시까지 시간별로 미끄럼틀/그네 사용량이 얼마나 나오는지 계산해준다.

void receive() 함수 : 아두이노로부터 데이터를 받을 때 자동으로 사용되는 함수이다. 인자로는 data와 ip값, port번호를 받는다. 수신한 데이터가 8100번 포트로 들어왔다면 미끄럼틀에 관한 데이터를 받은 것이다. 이 데이터를 insert문을 통해 db에 넣는다. 수신한 데이터는 미끄럼틀 id, case번호, startTime, 요일, duration 이다. /로 각 정보가 나뉘어서 들어오기 때문에 데이터를 수신한 후 '/' 문자 기준으로 데이터를 나눠줘야한다.(String dbString[] = message.split("/");)

db.query("insert into slides(₩"slides_id₩", ₩"case₩", ₩"startTime₩", ₩"day₩", ₩"duration₩") values("+dbString[0]+", " +dbString[1]+ ", ₩"" +dbString[2]+ "₩", ₩"" +dbString[3]+ "₩", "+dbString[4]+");");

수신한 데이터가 8200번 포트로 들어왔다면 그네에 관한 데이터를 받은 것이다. 이 데이터를 insert문을 통해 db에 넣는다. 수신한 데이터는 그네 id, case 번호, startTime, 요일, duration이다. /로 각 정보가 나뉘어서 들어오기 때문에 데이터를 수신한 후 '/' 문자 기준으로 데이터를 나눠줘야한다. (String dbString[] = message.split("/");)

db.query("insert into swing(₩"swing_id₩", ₩"case₩", ₩"startTime₩", ₩"day₩", ₩"duration₩") values("+dbString[0]+", " +dbString[1]+ ", ₩"" +dbString[2]+ "₩", ₩"" +dbString[3]+ "₩", "+dbString[4]+");");

UI

다음은 미끄럼틀의 관리자 화면을 구성하는 코드이다.

void setup() 함수 : UDP프로토콜을 통해 데이터를 받는 것을 명시한다. 8300번 포트로 미끄럼틀에 대한 데이터를 받는다.

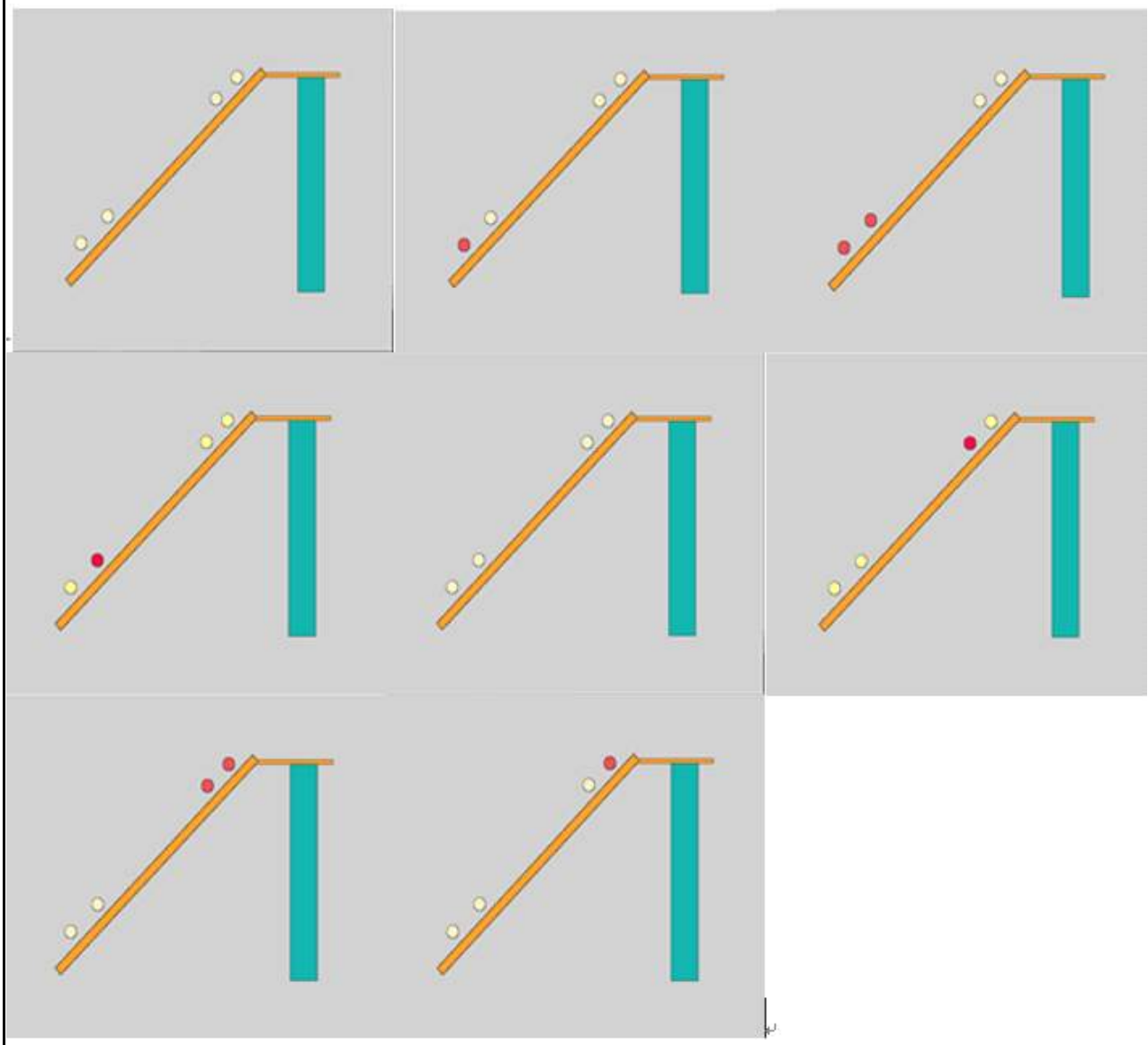
void draw() 함수 : 관리자에게 보여줄 미끄럼틀을 그리는 함수이다. 서버가 실행되자마자 자동으로 실행된다.

void draw1() 함수 : 아두이노에 있는 센서에 low값이 찍혔을 때(사람이 있다고 인식했을 때), 관리자 화면에 있는 센서가 켜지게 하는 함수이다. switch문을 통해 구현하였다.

void draw2() 함수 : 아두이노에 있는 센서에 high값이 찍혔을 때(사람이 없다고 인식했을 때), 관리자 화면에 있는 센서가 꺼지게 하는 함수이다.

void receive() 함수 : 아두이노로부터 데이터가 수신되면 자동으로 호출되는 함수이다. 데이터를 받아 split함수로 데이터를 정제한 후에 상황에 맞게 draw1()함수나 draw2()함수를 호출한다

<미끄럼틀 관리자 알림 화면>

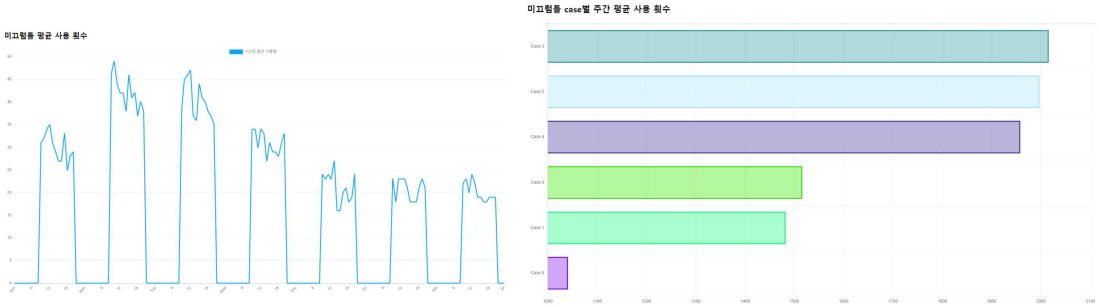


Webpage구현방법

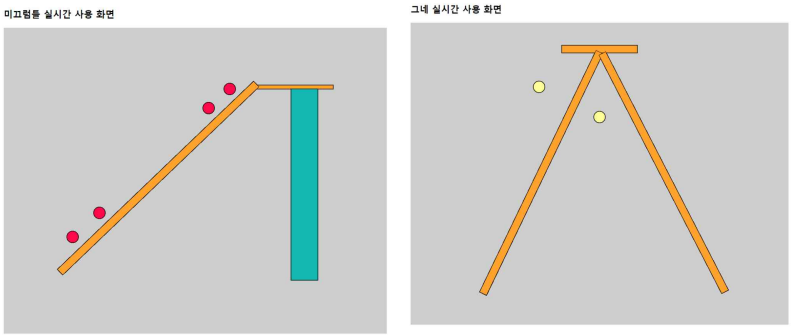
processing 서버에서 Webpage를 제공한다. 서버와 html은 post방식으로 데이터를 주고 받는다.

제공하는 정보는 놀이기구 이용정보 통계량, 실시간 이용정보이다. 놀이기구 이용정보 통계량은 요일과 시간별 평균 놀이기구 이용 횟수와 1주일당 case별 이용횟수를 보여준다. 요일과 시간은 일요일부터 토요일까지 0-24시이다. case별 이용횟수에서는 사용자가 많이 이용한 case 순위로 보여준다.

<결과화면>



실시간 이용정보에서는 미끄럼틀과 그네를 간략히 보여준다. processing에서 UI로도 보여주지만, 시스템을 전체적으로 보여주기 위해 webpage로도 보여준다. 미끄럼틀은 옆면을 보여주며 센서가 부착된 위치에 0표시를 한다. 사람이 없음면 노란색, 사람이 인식되면 빨간색이다. 그네는 옆면을 보여주며 센서가 있는 위치와 그네 바깥쪽 위치에 0표시를 한다. 그네 줄이 인식되면 빨간색, 인식되지 않으면 노란색이다. 그네가 높이 올라갔을 경우에는 그네 바깥쪽 0를 빨간색으로 바꾼다.



<코드 서술>

통계량은

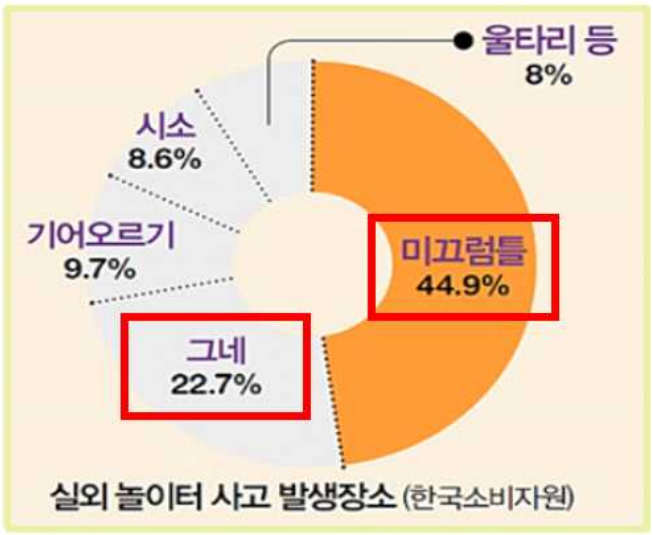
실시간 데이터 코드는 processing과 다르지 않다. javascript에서 swing이라는 객체를 만든다. swing에서는 canvas를 생성하고 UI의 코드와 같이 UI를 그린다. 단 receive와 draw1(), draw2()를 서버에서 처리하고 노란색과 빨간색에 대한 정보만을 JSON형태로 변환하고 post방식으로 html로 넘긴다. html에서는 넘어온 정보를 파싱하여 색상값으로 사용한다.

xhr.onload = function() : 그래프를 보여주는 화면에서는 서버로부터 받는 데이터가 있으면 그래프를 그린다.

let swing = function(p) : canvas를 만들고 미끄럼틀과 그네를 그리는 함수

3-4. 왜 지금인가?

옥외놀이터에서 일어날 수 있는 사고를 하나라도 막는 것이 저희의 목적입니다. 놀이터에서 신나게 놀다가 불의의 사고를 당하는 경우를 줄이기 위해 저희의 해결방안은 지금 구현되어야합니다. 한국소비자원의 통계자료에 따르면 실외 놀이터 사고 발생장소는 미끄럼틀과 그네가 67.6%를 차지합니다. 또한 행정안전부의 통계자료에 따르면 놀이터에서 이용자의 부주의로 인한 사고가 97.8%로 가장 많이 발생하는 사고 유형이었습니다. 이처럼 놀이터에서 이용자로 인한 미끄럼틀과 그네에서의 이용 사고가 가장 비번하게 발생하기에 하루 빨리 이에대한 대책 마련이 필요합니다.



《 사고원인별 사고현황 》 (단위 : 건)

구분	총 사고 건수	이용자 부주의	시설 결함	기타
계	286 (100%)	282 (98.7%)	3 (1.0%)	1 (0.3%)
주택단지	88	88	-	-
도시공원	19	18	1	-
유치원	27	27	-	-
학교	146	143	2	1
놀이제공영업소	3	3	-	-
아동복지시설	2	2	-	-
공공도서관	1	1	-	-

*** 자료 : 행정안전부 – 2017년 어린이놀이시설 안전사고 분석 결과**

3-5. 시장규모 및 시장지배력

사고 연령대별로 살펴보면, 활동량이 많은 학령기(7~14세) 어린이가 234명(79.6%) 으로, 취학전(1~6세) 어린이 57명(17.7%)보다 부상자 수가 약 4.1배 많음

출처, 행정안전부

KOSIS 국가통계포털
KOrean Statistical Information Service
보다 나은 정부
통계표

국내통계 | 국제·북한통계 | 쉽게 보는 통계 | 온라인간행

국내통계

※ "관심주제설정"에서 특정 주제를 선택한 경우

통계목록 | 현 거주지 성, 연...

1) 현 거주지 성, 연령 및 1년 전 거주지 유형별 인구(1세이상) - 시군구

자료갱신일: 2020-08-28 / 수록기간: 년 2016 ~ 2019 / 자료문의처 : 042-481-3756(전수), 042-481-3735(표본)

일괄설정 + | 항목[7/7] | 행정구역별(시군구… | 성별[3/3] | 연령별[2/17]

(단위 : 명) 새창보기 | 주석정보 | 주소정보 | 행렬전환 | 분석

행정구역별(시군구)	성별	연령별	2019 1년 전 거주지-계
전국	계	5~9세	2,251,867
		10~14세	2,219,445
	남자	5~9세	1,156,452
		10~14세	1,145,070
	여자	5~9세	1,095,415
		10~14세	1,074,375

행정안전부 조사에 따르면 79.6% + 17.7%= 97.3%로 1세~14세 어린이가 사고의 97.3%를 차지한다. 위의 표는 2020년에 갱신된 국가통계포털 자료로 전국의 5-14세 어린이의 인구수를 나타내고 있다. 이 자료에 의하면 서비스의 수혜자를 전국의 5~14세로 잡는다면 2,251,867+2,219,445=4,471,312명의 어린이로 잡을 수 있다.

또한 어린이의 부모(어린이 한명 당 2명)를 부수혜자로 잡는다면 대략 어린이의 부모인 4,471,312*2=8,942,624명으로 볼 수 있다.

총 수혜자는 4,471,312+8,942,624=13,413,936명이다.

4 기술성 및 사업화가능성

4-1. 기술 (제품, 서비스)의 기능 및 특징

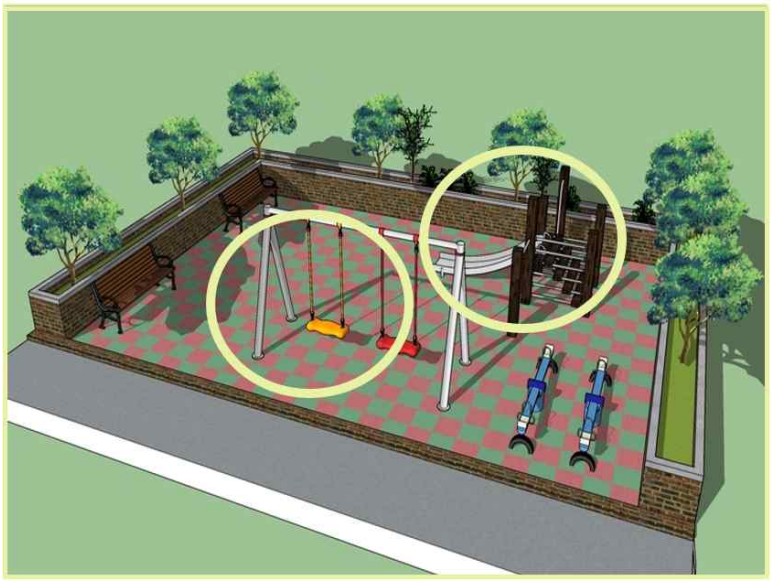
-사고 상황을 예방하기 위한 알림을 주어 사고발생률을 줄일 수 있다.
-보호자들이 안심하고 아이들을 놀이터로 내보낼 수 있다.
-놀이터 혼잡도에 대한 실시간 정보, 날씨 정보 등을 제공한다.
-사용자가 직접 놀이터에 가지 않고도 사용자에게 놀이터에 관한 정보를 미리 알려 주어 놀이터 이용에 도움을 줄 수 있다.
-놀이터의 시간에 따른 혼잡도, 날씨, 사고 이력 등의 정보들을 수집, 분석하여 통계를 낼 수 있다. 추후 이러한 데이터들을 활용하여 시스템을 확장시킬 수 있다.

4-2. 기존 경쟁기술(제품, 서비스)과의 개발기술(제품, 서비스) 차별성, 우위성, 응용성

관련 연구/서비스/시스템 조사 결과 및 한계점
기존에는 CCTV만을 사용하여 아이들의 안전을 확인하였고, 문제가 생기면 경비원이 와서 해결하였다. 실제로 아이들이 안전하게 노는지 지켜보는 것은 아이들과 함께 온 보호자의 몫이었다. 또한 키즈카페 등 실내놀이터는 안전요원이 있고 아이들을 항상 지켜보며 안전관리를 하지만 실외 놀이터는 그렇지 못하다.

4-3. 개발기술(제품,서비스)의 개략도

시스템 구성도- 전체 구성

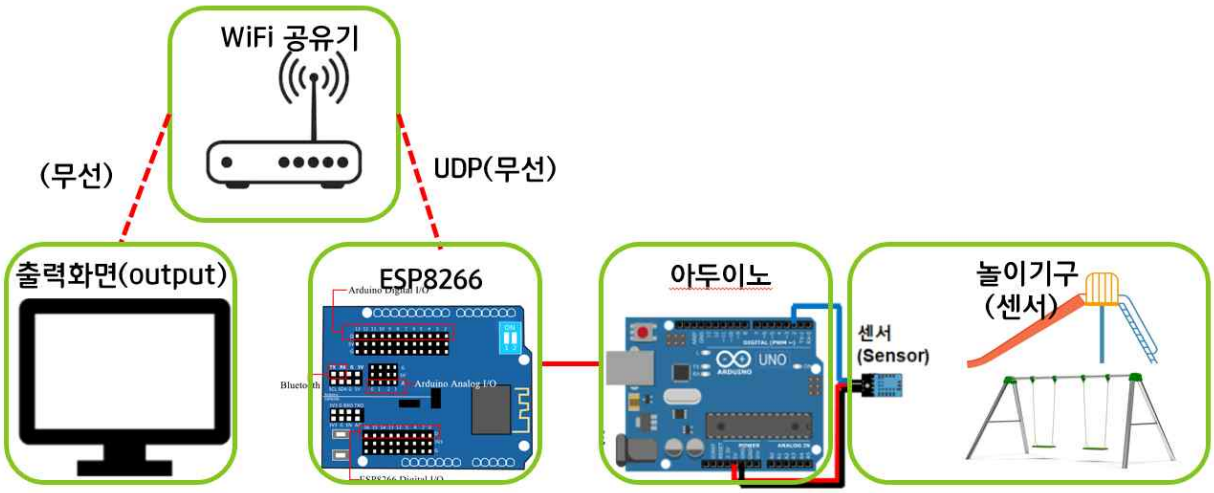


Target
놀이기구:미끄럼틀/그네
선정사유 : 대부분 놀이터에 있는 기본적인 놀이기구
장소 : 양천구에 있는 놀이터

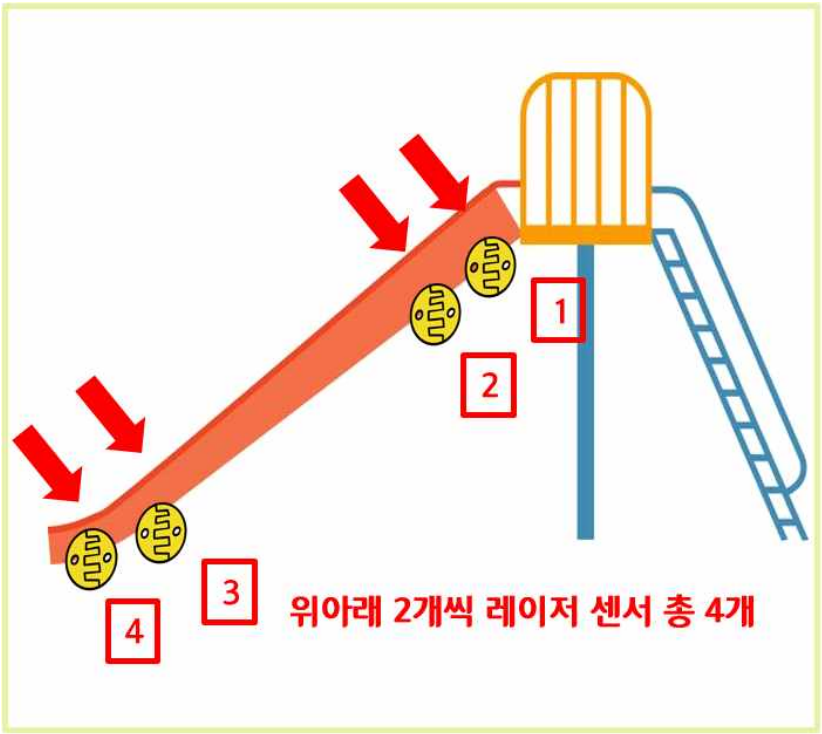
사용 센서 : 조도센서,
레이저(광)센서/트래킹 센서
통신 방식 :wifi(무선)
프로토콜 :UDP

언어: 아두이노 - C++
프로세싱 - JAVA
DB: SQLite

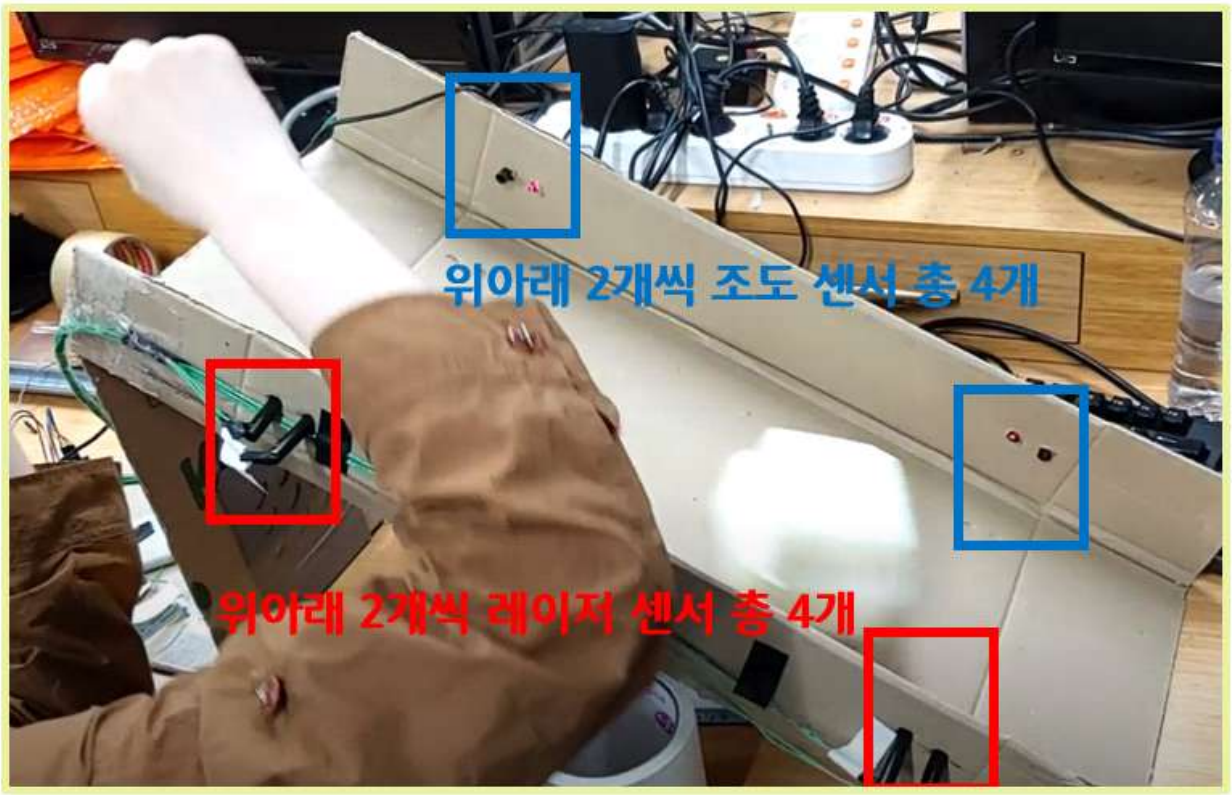
시스템 구성도- 통신



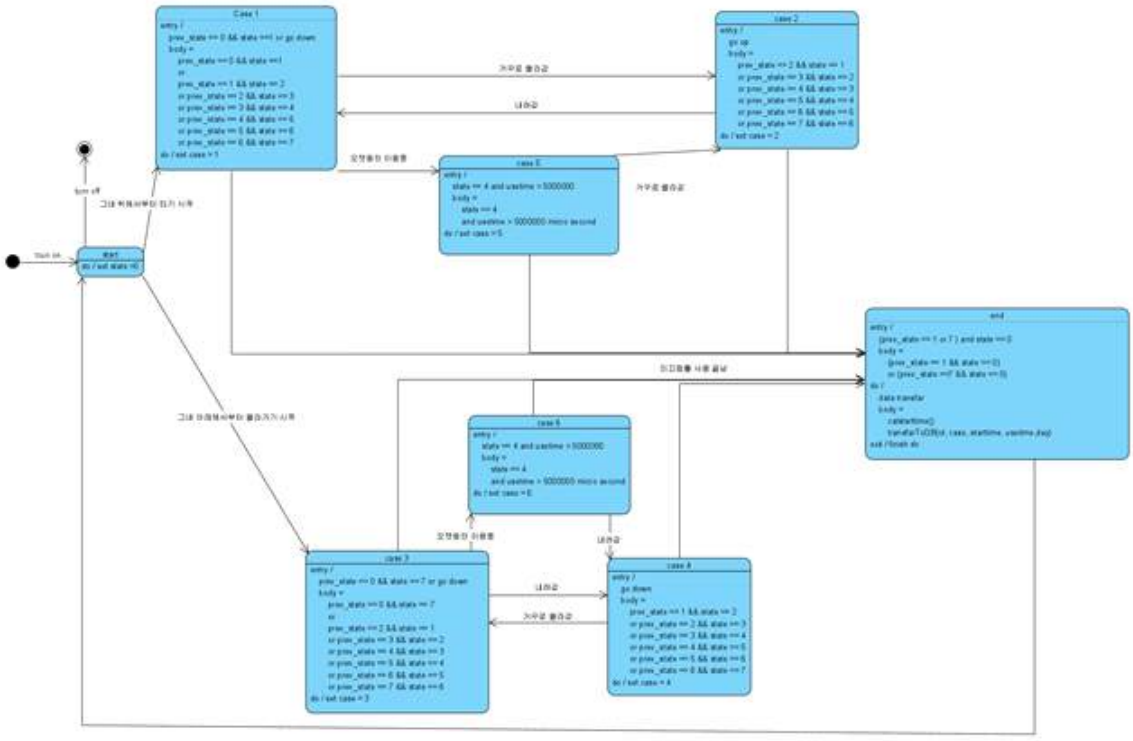
미끄럼틀 구조

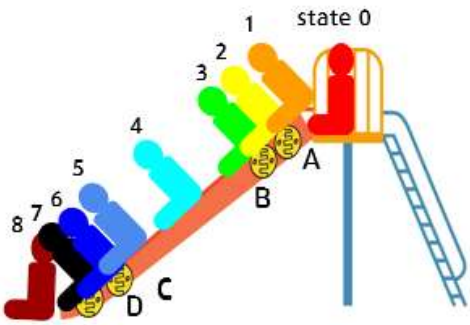


미끄럼틀 모형 사진



미끄럼틀 state diagram



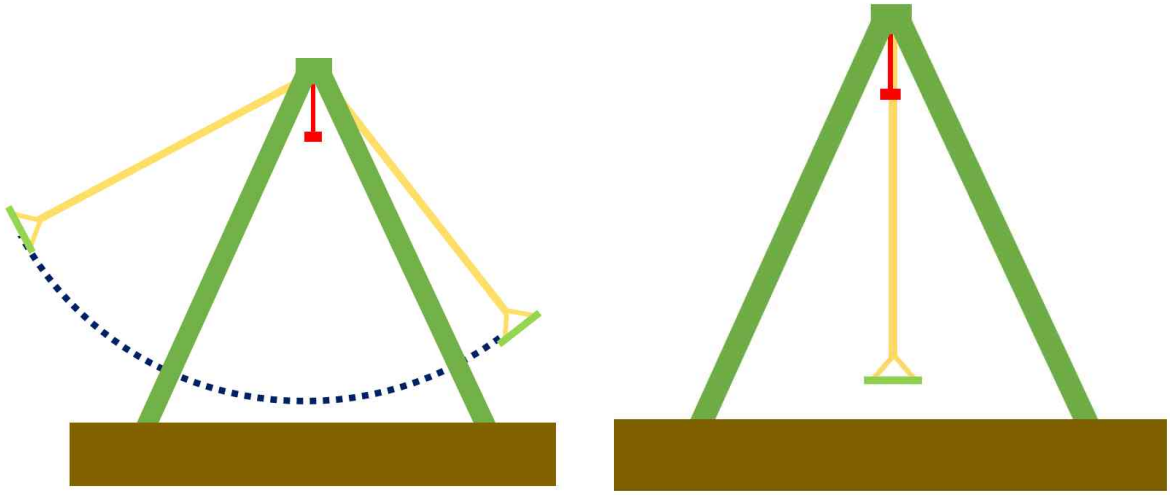


case	상황
1	순서대로 내려옴
2	위에서, 내려갔다 올라옴
3	거꾸로 올라옴
4	밑에서, 위로 올라갔다 내려옴
5	순서대로 내려옴& 중간에 머무름
6	거꾸로 올라옴 & 중간에 머무름

미끄럼틀 설명

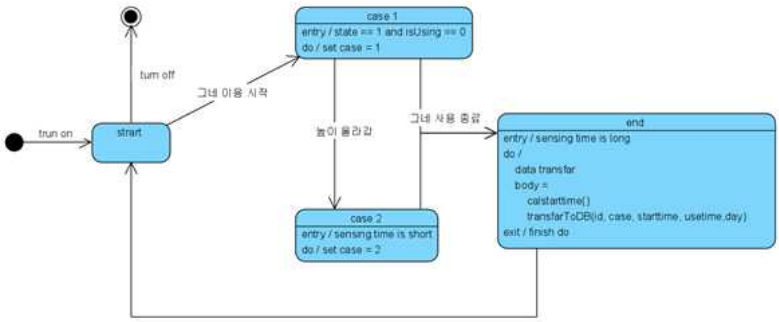
미끄럼틀 위에나 바닥에 있을 때는 Start 또는 End, A 센서만 가리면 1번, A,B센서를 가리면 2번, B 센서만 가리면 3번, B랑 C 사이에 있으면 4번, C 센서만 가리면 5번, C와 D센서를 가리면 6번, D 센서만 가리면 7번입니다.
선으로 표현되는 부분은 상태가 변화하는 부분입니다.
상태가1->2->3->4->5->6->7로 내려가거나 또는
7->6->5->4->3->2->1로 올라가거나, 중간에 오래 멈춰있으면 변화합니다.
Case 1만 설명하자면 Starte 상태에서 아이가 A센서에 인식되면 1번 상태가 되어 Case 1이 됩니다. Case 1은
1->2->3->4->5->6->7로 내려가야하는 케이스인데 중간에 4->3으로 올라가는 현상이 발생한다면 Case 2로 바뀝니다. 또한 State가 4이면서 오래 머무르면 Case 5로 바뀝니다.

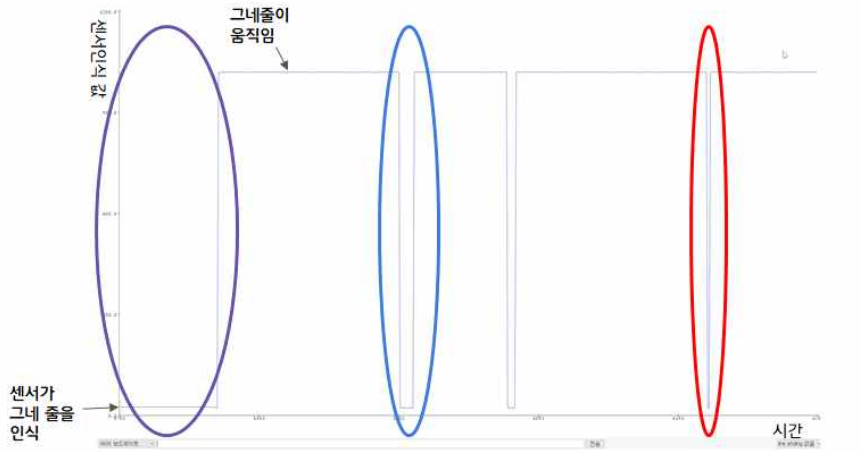
그네 구조



그네 모형 사진

그네 state diagram





그네 설명

그네의 상태도 입니다.
아두이노가 작동을 시작하면 시스템이 시작합니다. state는 센서가 인식하는 값입니다. state가 1이고 그네가 사용중이 아니면 그네 이용이 시작된다고 보고 case를 1로 설정합니다.
그네를 이용하다가 높이 올라가서 그네 줄이 센싱 된 시간이 매우 짧으면 case를 2로 바꿉니다.
그네 줄이 sensing되는 시간이 길면 그네가 멈춘것으로 보고 DB에 데이터를 보내고 start로 돌아갑니다.

그래프에서 가로축은 시간, 세로축이 센서인식값입니다. 보라색 구간은 사용자가 없어 그네줄이 계속 인식되는 부분입니다. 그네가 움직이면 센서에 줄이 인식되지 않아 high값이 입력되는 것을 볼 수 있습니다. 그네가 움직일 때 그네 줄이 센서 앞을 지나가면서 파란색 영역과 같이 잠깐동안 low값이 인식 됩니다. 사용자가 그네를 너무 높게 타면 빨간색 부분과 같이 low값이 인식되는 영역이 기존보다 매우 짧게 나타납니다. 이를 통해 사용자가 그네를 빠르고 높게 타고 있는 것을 알 수 있습니다.

5 성장성 및 시장경쟁력

5-1. 국내·외 관련기술(제품,서비스) 동향, 경쟁사 현황

실내 놀이터와 달리 옥외 놀이터는 안전관리에 대한 서비스들이 제공되지 않고 있음을 깨닫고 이 문제를 해결하고자 노력하였습니다. 우리가 제시한 서비스의 주제와 관련된 기존의 서비스가 없음을 확인하였습니다. 유사서비스를 제공하는 경쟁사가 없다는 점에서 강점을 가지고 있습니다.

관련 서비스의 발전으로는 센서 종류과 수집데이터의 추가입니다. 공공 기상 데이터가 아닌 온습도 센서와 조도 센서를 통하여 놀이터에서 직접 측정하는 것으로 놀이터 지역범위만의 정확한 온도, 습도, 조도 파악이 가능합니다.

온습도 센서를 이용하여 놀이터의 온도와 습도를 측정

조도 센서를 이용하여 놀이터의 조도를 측정

초음파 센서를 이용하여 어린이의 접근 여부 파악, 미끄럼틀의 밑부분, 윗부분에 어린이가 접근하고 있는지 여부가 파악 가능합니다.

5-2. 시장의 성장성 및 진입가능성

예상되는 고객과 판매처는 도시를 관리 담당하는 공공기관과 사설 시설 관리 회사입니다. 공공기관의 경우 공공놀이터 시설, 사설 관리 회사의 경우 아파트나 회사 안에서 운영되는 사설 공공놀이터 시설에 서비스가 적용될 수 있습니다.

경쟁회사는 아니지만 놀이터라는 공통분야의 회사로 (주)에넥스트가 있습니다. 다음의 기사를 보면 대학생과 에넥스트가 계약을 체결하여 해당 아이디어 제공자가 서비스에대한 로열티를 발급받고 있습니다. (한남대학교 김완기 학생의 경우 최근 국내 어린이 놀이기구 및 야외 운동기구 생산업체인 '에넥스트'와 로열티 계약을 체결해 미래 한국을 이끌 우수 디자이너로서 주목을 받았다.) 이런 방식으로 놀이터 시설 생산, 관리 업체와 계약하는 방법도 있습니다.
https://news.mt.co.kr/mtview.php?no=2016080910065562967&outlink=1&ref=http%3A%2F%2Fsearch.naver.com

기존에 이러한 비슷한 서비스를 제공하고 있는 경쟁회사가 없기 때문에 서비스경쟁력으로 인해 상대적인 가격경쟁력 또한 발생합니다.

5-3. 판매전략 및 마케팅 방안

영상을 통해 실제 설치, 시현모습을 보여줌으로서 홍보와 판매를 합니다. 서비스를 판매함으로 수익을창출 합니다. 학부모를 대상으로 한 홍보를 통해 사업의 지지를 얻을 수 있습니다.
시장에서 성공하기 위한 결정적인 요소는 사고발생률을 얼마나 줄이느냐에 달려있다고 생각합니다. 이 시스템으로 인해 사고발생률을 줄이면 시스템에 대한 사용자의 신뢰도가 높아지고 자연스럽게 홍보가 될 것이라고 생각합니다. 또한 올해 정부의 주요 이슈 중 하나가 안전인만큼 정부의 지원을 받아 홍보하고 수익을 창출할 예정입니다.