

**-Technical writing-**

**-Contents-**

**Ⅰ. Topic**

* *topic*
* *Schematics*

**Ⅱ. Hardware**

**Ⅲ. Software**

*- Main Camera*

MainCamera\_py2.py / touch\_window.py / Main\_filter.py /

Contours.py

*- Keyboard*

Keyboard\_Controller.py

*- Lidar*

Lidar.py

*- Control server*

Views.py

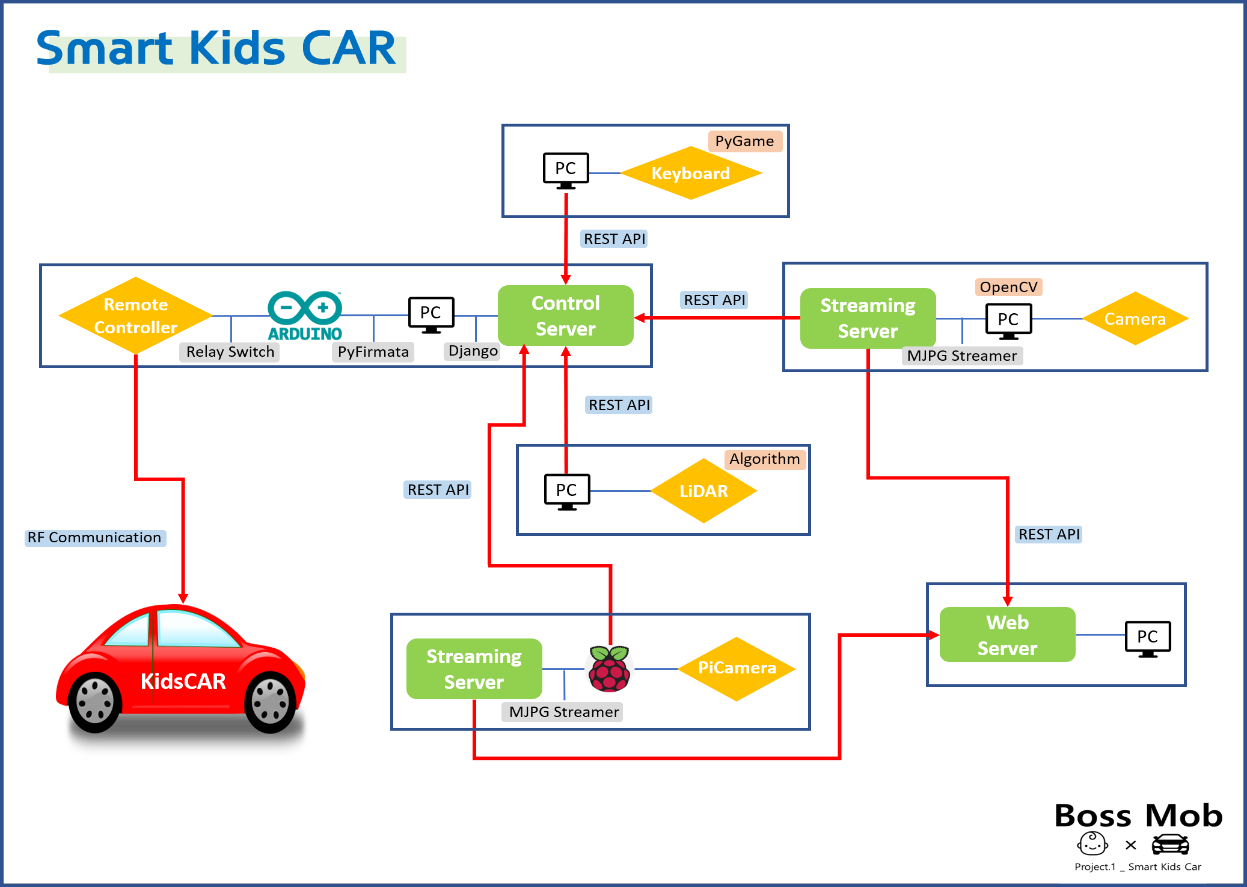
*- Web server*

Views.py / run.html / baby.html

**- Topic -**



부모의 편리와 자녀의 안전을 보장하는 놀이 환경을 제공하기 위해 안전구역 설정, 충돌방지, 실시간 모니터링 기능을 구현함. 이 시스템을 통해 어린 자녀가 있는 가정에 즐거운 여가생활을 제공할 수 있을 것으로 기대함.

**<Schematics>**

**- Hardware -**

**1. 전동차:** [대호토이즈] 아우디 R8 스파이더 유아전동차 12V 듀얼모터

**2. Remote controller:** 12V 전동차 조종기 – J(타입) SHSB12VWS-3

**3. Arduino UNO**

**4. Relay module:** 아두이노 8채널 5V 릴레이 모듈 [SZH-RLBG-009]

**5. Raspberry pi**

|  |  |
| --- | --- |
| Raspberry pi | Raspberry pi 3B+ |
| Micro SD card | MicroSDHC 32GB, Class10, EVO Plus, UHS-I(U1)  [SD어댑터포함] [MB-MC32GA/KR] |
| 방열판 | 방열판 세트 [SZH-AT009] |
| Pi camera cable | 라즈베리파이 터치스크린/카메라용 1mm피치 15p FFC 리본케이블 50cm |
| Pi camera | raspberry pi camera V2 |

**6. Web camera:** 로지텍 HD 웹캠 C270

**7. 삼각대:** 아이크라운 스마트폰 삼각대

**8. PC(3대), Tablet(2대), Raspberry-pi(1대)**

(1) PC: mjpg-streamer, OpenCV / web server, keyboard control / control server

(2) Tablet: touch window, OpenCV / Lidar /

(3) raspberry pi: mjpg-streamer

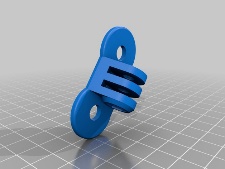
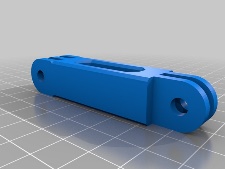
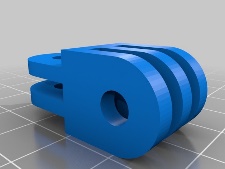
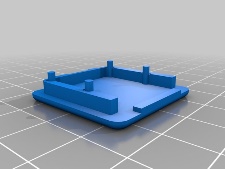
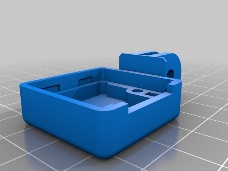
**9. LiDAR:** 60 Degree Laser Scanner Development Kit - RPLIDAR A1M8 [DFR0315]

**10. 3D프린팅 부품(pi camera mount, lidar 받침대)**

(1) pi camera mount

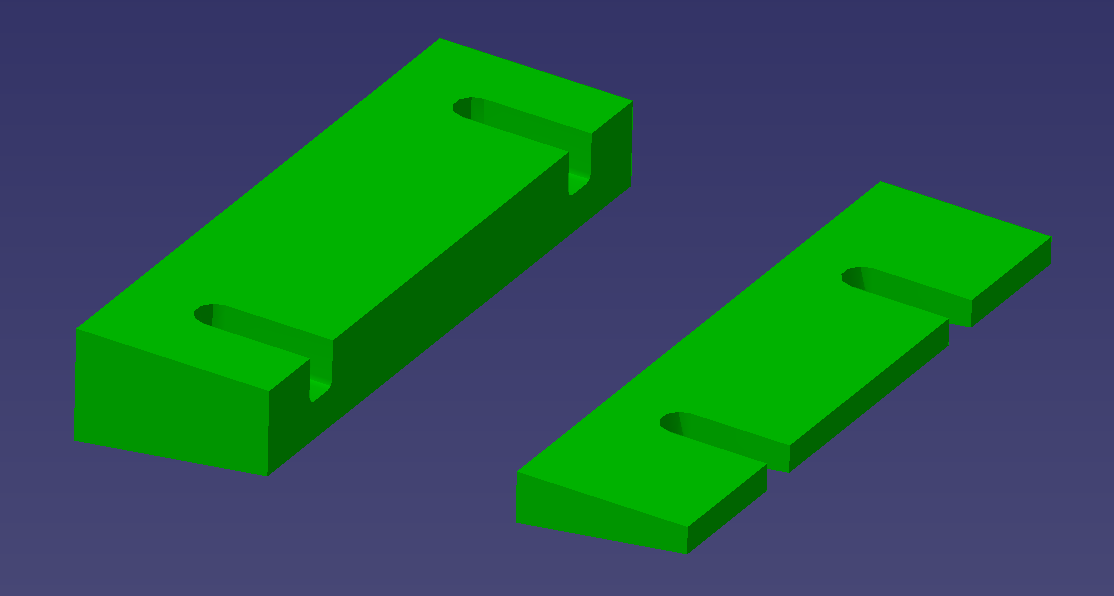
<https://www.thingiverse.com/> : 3d 모델 파일 공유 사이트

raspberry pi mount, pi camera mount 각각 검색 🡪 stl 파일 다운로드 🡪 Cura, Slic3r, Skeinforge 소프트웨어를 통해 gcode 변환하여 출력

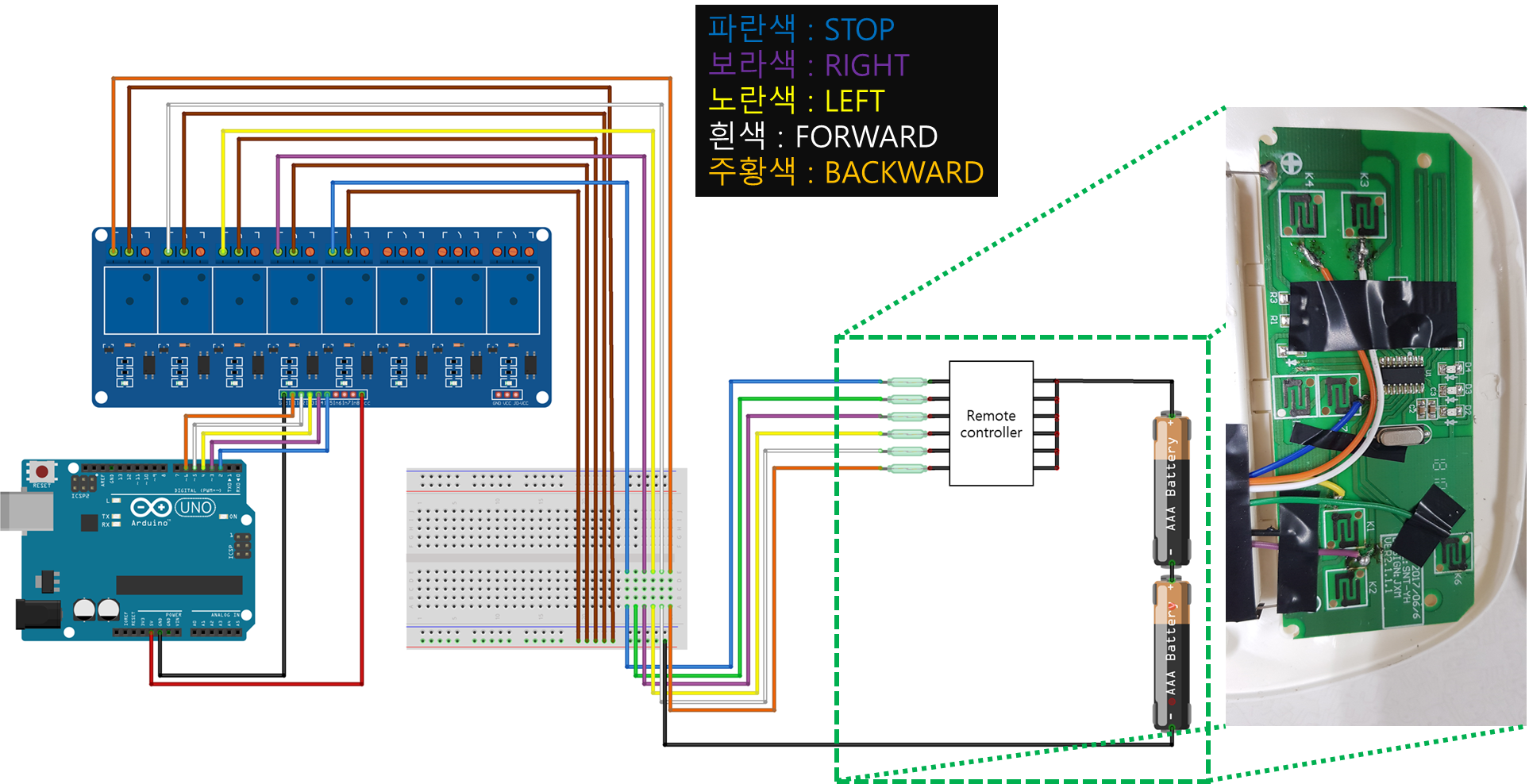


(2) lidar 받침대

lidar 규격 측정 🡪 CATIA를 통해 stl 파일로 저장 🡪 gcode 변환 후 출력



**11. remote controller 배선**



**- Software -**

**<Software Architecture>**

|  |
| --- |
|  |

**MainCamera\_py2.py**

경로 : Smart\_Kids\_Car\mjpg\_ver4\mjpg\mjpg-streamer\mjpg-streamer-experimental

메인 카메라를 작동시킬 때, 실제로 실행시키는 파일. 메인 카메라의 스트리밍의 시작, 종료를 제어함. 사용자 디바이스의 touch\_window.py 파일과 정보를 주고받으며, 사용자 디바이스와 소켓통신을 통해 스트리밍 시작과 종료 신호를 주고 받음. 같은 폴더안의 start\_uvc.sh 파일과 start\_opencv.sh 파일을 통해 스트리밍을 시작하게 되어있음.

1. **Dependencies:**

Python2, subprocess, socket, time

1. **Global Variables**

Socket\_PORT : 메인 카메라에서 소켓통신에 사용하는 포트 번호

1. **Functions**
2. **Request**
3. **Remarks**

카메라 경로 설정 : start\_uvc.so, start\_opencv.so 파일내부의 input 설정 부분(-i ~)에서 카메라 경로 설정 부분(-d ~ )을 수정하여 어떤 카메라를 사용할 지 설정할 수 있음. 웹캠의 경우 dev/video0 또는 dev/video1로 설정하면 됨.

**touch\_window.py**

경로 : Smart\_Kids\_Car\touch\_window

사용자 디바이스에서 Safety zone과 Tracking 할 Object를 지정하고, 메인 카메라로 전송해주는 파일. send.bat(scp 통신) 파일을 통해 Safety zone과 Object 정보(.npy 형태)를 메인 카메라로 전송하며, 메인 카메라와 소켓 통신을 통해 스트리밍 시작과 종료 신호를 주고받음. 최종적으로는 Webapp을 팝업시킴.

1. **Dependencies:**

Python3, OpenCV(4.1.0), numpy, socket, time, webbrowser.

1. **Global Variables**

Server\_IP : 메인 카메라의 IP

Socket\_PORT : 메인 카메라에서 소켓 통신에 사용하는 포트 번호

Web\_URL : 웹 서버의 URL

1. **Functions**

def **mouse** (event, x, y, flags, param)

mouse event에 따라 실행되는 mouse call back 함수. 마우스 왼쪽 버튼을 누를 때, 뗄 때, 그리고 마우스 움직임이 발생될 때 실행된다. 마우스 클릭 횟수에 따라 다른 기능들이 실행되도록 하였음.

첫 번째 클릭 : Safety zone 설정모드 진입

두 번째 클릭과 동시에 드래그 : Safety zone 설정

세 번째 클릭 : Tracking Object 설정모드 진입

네 번째 클릭과 동시에 드래그 : Tracking Object 설정

Local Variables

Return

1. **Request**

Web app 접속 : Web\_URL + “/?action=landing”

1. **Remarks**

send.bat 파일에 메인 카메라의 IP와 Safety zone과 Object 정보(.npy 형태)를 저장할 경로, 그리고 사용자 디바이스에서 touch\_window.py 파일의 경로를 입력해주어야 함.

**Main\_filter.py**

경로 : Smart\_Kids\_Car\mjpg\_ver4\mjpg\mjpg-streamer\mjpg-streamer-experimental\plugins\input\_opencv\filters\cvfilter\_py

Mjpg-streamer에 의해 실행되는 파일이며, 실질적인 Object Tracking과 Object가 Safety zone에서 벗어났는지를 판단하는 파일.

1. **Dependencies:**

Python2, OpenCV(3.4.2), numpy, requests

1. **Global Variables**

streaming\_IP : 웹 서버의 IP

streaming\_Port : 웹 서버를 구동 하는 컴퓨터에서 서버를 여는 포트 번호

control\_IP : 컨트롤 서버의 IP  
control\_PORT : 컨트롤 서버를 구동 하는 컴퓨터에서 서버를 여는 포트 번호

initBB : Tracking 할 Object

initFrame : Tracking Object를 설정하는 순간의 프레임

contour : Safety zone의 좌표(contour 형태)

pts\_n : Safety zone의 좌표(numpy array 형태)

tracker : Tracker 객체(CSRT 알고리즘 사용)

1. **Functions**

def **init\_filter** ()

mjpg-streamer(input\_opencv.so)가 실행 될 때, 단 한번만 실행되는 함수. 사용자 디바이스에서 전송한 Safety zoner과 Object 정보를 불러들여 변수에 저장함. Object Tracker(CSRT)를 초기화 함.

Local Variables

Return

def **process** (self, img)

mjpg-streamer(input\_opencv.so)가 실행되는 동안, 매 프레임마다 영상처리(Object Tracking, Safety zone 벗어났는지 판단)를 하며, Safety zone을 나갔는지 여부를 컨트롤 서버와 웹 서버에 REST API 형태로 전송함.

Local Variables

Return: 영상처리 된 이미지를 반환

1. **Request**

Safety zone IN : "http://" + streaming\_IP + ":" + streaming\_Port + "/?action=" + “in”

Safety zone OUT : "http://" + streaming\_IP + ":" + streaming\_Port + "/?action=" + “out”

Safety zone IN : "http://" + control\_IP + ":" + control\_PORT + "/Main\_cam/" + "I"

Safety zone OUT : "http://" + control\_IP + ":" + control\_PORT + "/Main\_cam/" + "O"

1. **Remarks**

Line4(sys.path.insert(~))에서 Contour\_py2.py의 경로를 추가해줘야 함.

**Contours.py**

Contours\_py2 경로 : Smart\_Kids\_Car\mjpg\_ver4\mjpg\mjpg-streamer\mjpg-streamer-experimental

Contours\_py3 경로 : Smart\_Kids\_Car\touch\_window

사용자가 그린 안전구역(cv2.polylines)을 배경과 분리하고 cv2.findContours로 좌표화 하여 cv2.pointPolygonTest를 이용해 Tracking box(전동차)가 안전구역 내부에 있는지 판단함.

1. **Dependencies:**

numpy, cv2

1. **Global Variables:**
2. **Functions:**

def **makeContour** (img,pts\_n)

for문을 사용하여 안전구역이 그려진 이미지의 픽셀 중 안전구역 색상 BGR(255,0,0)은 흰색BGR(255,255,255)로 하고 그 외 색상은 검정색 BGR(0,0,0)으로 함. 그 후 안전구역 내부를 BGR(255,255,255)로 채우고 흑백으로 변환한 후 이진화 함. 그 후 안전구역을 좌표화 함.

Local Variables

**img\_fill**: cv2.fillPoly로 안전구역 내부를 흰 색으로 채움

**img\_gray**: cv2.cvtColor로 img\_fill을 흑백으로 함

**img\_threshold**: cv2.threshold로 img\_gray를 이진화 함

**img\_contour**: cv2.findContours로 img\_threshold를 좌표화 함

Return: img\_contour

def pointTest(p,contour)

좌표화 된 안전구역과 아이와의 최소거리를 실시간 계산하여 그 값이 양의 값을 갖는지 음의 값을 갖는지 판단

Local Variables

dist : cv2.pointPolygonTest로 아이의 위치 판단

Return: True (dist>=0인 경우), False (dist<0인 경우)

1. **Request**

**Keyboard\_Controller.py**

경로 : Smart\_Kids\_Car\Keyboard\_Controller

아이의 전동차 조작을 대신하여 키보드 조작을 가능케 함. 직진, 후진, 좌회전, 우회전, 긴급정지 조작 가능. while문을 사용하여 사용자가 어떤 키를 누르는지 혹은 눌렀다 떼는 지를 입력 받고 이를 변수 “cmd”로 저장하여 control server로 requests를 보냄.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 키보드 | w | s | a | d | b | Q |
| cmd | F | B | L | R | S | q |
| 기능 | 전진 | 후진 | 좌회전 | 우회전 | 긴급정지 | 모든 키 신호 low |

|  |  |  |
| --- | --- | --- |
| 키보드 | 눌렀던 w, s, b를 뗀 경우 | 눌렀던 a, d,를 뗀 경우 |
| cmd | M | N |
| 기능 | 전진, 후진, 긴급정지 신호를 low로 함 | 좌회전, 우회전 신호를 low로 함 |

1. **Dependencies:**

pygame, time, requests

1. **Global Variables**

**control\_IP**: control server의 IP입력

**control\_PORT**: control server의 PORT번호 입력

1. **Functions:**

def **send** (cmd)

REST API를 사용하여 키보드 입력을 다음의 형태로 Control server로 전송함. requests.get("http://" + control\_IP + ":" + control\_PORT + "/keyboard/" + cmd)

Local Variables: -

Return: -

1. **Request**

전진(F): <http://169.254.202.202:8000/keyboard/F>

후진(B): <http://169.254.202.202:8000/keyboard/B>

좌회전(L): <http://169.254.202.202:8000/keyboard/L>

우회전(R): <http://169.254.202.202:8000/keyboard/R>

긴급정지(S): <http://169.254.202.202:8000/keyboard/S>

모든 키 신호 low(q): [http://169.254.202.202:8000/keyboard/q](http://169.254.202.202:8000/keyboard/q/)

전진, 후진, 긴급정지 신호 low(M): <http://169.254.202.202:8000/keyboard/M>

좌회전, 우회전 신호 low(M): <http://169.254.202.202:8000/keyboard/N>

1. **Remarks**

본 파이썬 파일은 키보드 입력을 받는 while문으로 이루어져 있으며 pygame 모듈을 이용하여 사용자의 키보드 입력을 받음. 키보드 입력은 키를 누르는 경우, 뗀 경우로 나눠 받으며 이는 “flag” 변수를 true, false로 각각 지정하여 if문으로 구분한다. 입력 받은 키는 “buttons” 변수로 저장되면 각각의 경우에 맞춰 선언된 “send” 함수를 이용하여 control server에 REST API로 requestsget 한다.

**Lidar.py**

경로 : Smart\_Kids\_Car\Lidar

Lidar setting 및 차량의 예상경로를 생성함. 이후에 lidar가 차량 주변의 장애물을 스캔하고, 이를 예측경로와 비교하여 장애물의 위치에 따라 알고리즘 case별 차량 제어신호를 control server로 보냄.

1. **Dependencies:**

rplidar, numpy, scipy, matplotlib, requests

1. **Global Variables**

**control\_IP** : request를 보낼 대상인 control server의 IP 주소

**control\_PORT** : control server의 PORT 번호

**PORT\_NAME** : lidar를 연결한 USB PORT 이름

**path parameters** : 차량의 예상경로 좌표화 및 interpolation을 위한 변수모음

**tuning parameters** : lidar의 감지범위, 정지거리, 노이즈 등을 조절하는 변수모음

**check parameters** : 예상경로 위에 장애물 존재여부를 표현하는 변수모음

**currentmode** : 차량의 현재 속도를 저장하는 변수

**cmd** : 알고리즘 판단 결과에 따라 차량을 제어할 신호를 저장하는 변수

1. **Functions**

def **send** (cmd)

차량을 제어할 신호가 저장된 cmd 변수를 REST API 형태로 request를 control server에 보내는 함수임.

Local Variables

Return

def **isfront\_left, \_right, \_center** (x,y,theta)

lidar에서 감지한 장애물이 차폭을 고려하여, 차량의 예상경로(좌, 우, 중앙) 중 어느 경로 위에 존재하는지를 구분하는 함수임.

Local Variables

**x, y** : lidar에서 감지한 장애물의 x, y 좌표

**theta** : lidar에서 감지한 장애물의 각도[degree] **noise** : lidar 감지 결과가 장애물인지 noise인지 비교하는 변수

Return

**check** : 장애물이 경로위에 겹치는지 여부에 따라 1 or 0을 저장

def **setspeed** (mode)

차량의 현재 속도(mode)를 확인하여 1단 혹은 3단으로 변경시켜주는 함수임.

Local Variables

**mode** : 차량의 현재 속도를 확인하기 위한 변수

**currentmode** : 차량의 속도 변경 결과를 업데이트하고 저장함.

Return

**speedsignal** : 차량의 속도를 몇 회 변경시킬지 기억하는 변수

def **update\_scan** (num, iterator, line)

lidar에서 실시간으로 수집하는 scan 결과를 필터링하고, 동시에 isfront\_ 함수들을 거쳐 경로상에 존재하는 장애물의 위치에 따라 각 알고리즘 case에 부합하는 제어 신호를 control server로 보내주는 함수임.

Local Variables

**scan\_data** : lidar의 360도 스캔결과를 필터링하여 다시 저장함.

**oxcheck** : 알고리즘 case를 기억하고, 변경될 때에만 업데이트됨.

Return

**cmd** : 알고리즘 case에 따라 차량을 제어할 신호를 저장함.

1. **Request**

"http://"+control\_IP+":"+control\_PORT+"/lidar/"+cmd : 차량 제어신호(cmd)를 control server로 보냄.

**Views.py(Control server)**

경로 : Smart\_Kids\_Car\Control Server\Control Server\newapp

control server에서 전동차 제어를 위한 코드로 Main camera, lidar, keyboard로부터 받은 requests에 따라 아두이노 리모컨 신호를 제어함.

1. **Dependencies:**

django, pyfirmata, time, requests, threading

1. **Global Variables:**

**SignalForward(Backward, Right, Left, Brake)**: 리모컨 제어 아두이노 핀 번호**IOflag:** main camera로부터 판별된 전동차의 in, out을 구분하기 위한 flag

**lidarflag:** lidar에서 s, r, l의 requests를 받으면 라이다 제어를 우선 순위로 함

**adflag:** lidar에서 a, d의 requests를 받으면 해당 알고리즘 실행을 위한 flag

**prev\_dir, prev\_ver:** 자동복귀 기능을 위해 안전구역 나가기 직전 조작을 저장함

**cmd\_M:** main camera에서 REST API로 받은 in 또는 out 판정이 저장됨.

**cmd\_L:** lidar에서 REST API로 받은 장애물 위치에 관한 판정이 저장됨

**cmd\_K:** keyboard에서 REST API로 받은 전동차 조작 명령이 저장됨

**counter:** 키보드 threading 시작을 위한 변수

**Exit:** 리모컨 제어 아두이노 신호를 모두 low로 함

1. **Functions:**

def **Main\_Cam** (requests, cmd)

main camera에서 받은 requests(cmd\_M)에 따라 IOflag를 0(In) 또는 1(Out)로 함

Local Variables

Return: HttpResponse(cmd)

def **lidar** (requests, cmd)

lidar에서 받은 requests(cmd\_L)에 따라 lidarflag, adflag, aflag, dflag를 바꿔주어 함수 “def control”에서 case별로 전동차 제어를 가능케 함.

Local Variables

Return: HttpResponse(cmd)

def **keyboard** (requests, cmd)

keyboard\_controller\_server.py 실행을 위한 threading 실행과 lidar와 keyboard 조작 간의 우선순위를 세움.

Local Variables

Return: HttpResponse(cmd)

def **control** ( )

전동차 제어를 위해 아두이노(리모컨)와 pyfirmata를 통한 serial통신을 시작하며, main camera, lidar, keyboard에서 보낸 requests에 따라 나눠진 case 별로 아두이노 핀에 0 또는 1의 신호를 줌.

|  |  |  |  |
| --- | --- | --- | --- |
| IOflag = 0 (IN) | | | IOflag = 1 (OUT) |
| lidarflag = 0 | | lidarflag = 1 | prev\_dir과 prev\_ver에 따라 case별로 다르게 안전구역 내부로 자동 복귀 |
| adflag = 0 | adflag = 1 | Lidar에서 받은 s, l, r에 따라 각각 긴급정지, 좌회전, 우회전으로 회피 |
| Keyboard로 모든 조작 가능 | Lidar에서 받은 a, d에 따라 각각 좌회전과 우회전 조작을 제한 |

Local Variables: -

Return: -

1. **Request**

**Views.py(app.)**

경로 : Smart\_Kids\_Car\finalserver3\finalserver3\mysite\elections

Django frame work에서 사용하는 view file로 web application을 지원하는 host server에서 받는 request에 따라 client server에 어떤 context와 template을 response 할 지 정함.

1. **Dependencies:**

HttpResponse, threading, **face**, requests, json, time

1. **Global Variables**

**Happiness, anger, neutral** : azure face api에서 받아오는 아이의 감정상태 값

**Subscription Key** : azure service 구독 인증 키 값

**Image\_url** : baby cam에서 받아오는 snapshot의 url 주소 azure api의 request

**Message** : 안전 구역 경계선에서 전동차량이 벗어날 경우 out 신호를 표시함.

**T** : azure api 감정인식을 5초마다 수행해주는 thread를 의미함.

1. **Functions**

def **emotion** (happiness, anger, neutral)

감정인식을 하는 함수로 thread t의 target으로 설정하여, django server를 돌리는 동시에 5초 간격으로 아이의 감정을 분석함. /?aciton = emotion request로 실행함.

Local Variables

response : request.post로써, face api server의 response를 받음.

param : response를 받고 싶은 항목을 지정해주는 변수

Return : happiness, anger, neutral

def **index** (request)

view file의 index 함수로 request를 받아 rest api를 처리하는 역할음 담당하는데 이를 통해 client는 원하는대로 web app page를 request를 통해 제어할 수 있음.

Local Variables

Return

1. **Request**

<https://hostip/?action=main>

<https://hostip/?action=baby>

<https://hostip/?action=run>

<https://hostip/?action=emotion>

<https://hostip/?action=out>

<https://hostip/?action=in>

**run.html**

경로 : Smart\_Kids\_Car\finalserver3\finalserver3\mysite\elections\templates\elections

Django에서 사용하는 template file로 view에서 받는 request 중에서 <https://hostip/?action=run>을 통해 받아오며, 사용자가 main cam과 baby cam에서 받아오는 영상을 한 번에 볼 수 있으며, 경계구역 알람 기능이 탑재되어있음.

1. **Dependencies :**

j-quary, javascript, html2canvas, requests, bootstrap, **static**

1. **Global Variables**

**timerid1** : 경계구역 이탈 여부를 0.3초 간격으로 체크하는 역할과 알림기능 담당.

**timerid2** : 안전구역 자동 복구 기능의 정상 작동 여부를 0.3초 간격으로 체크함.

**timerid3** : 경계구역 이탈 시 main.html로 이동하여, 사용자에게 편의기능 제공.

**timerid7** : azure api를 통해 아이의 감정 상태를 5초 간격으로 확인하여 알려줌.

1. **Functions**

function **mmm** (a)

timerid1에 대한 javascript 함수로 0.3초 간격으로 message context를 업데이트하여 out request시에 알림음과 메세지 그리고 자동 페이지 넘김 설정 기능을 담당.

Local Variables

**a** : document.getElementbyid("logo”).innerText로 {{message}}

Return : #name h3, #move, #playsound, #start2, #stop1

function **mmm2** (a2)

timerid2에 대한 javascript 함수로 0.3초 간격으로 safe context를 업데이트하여 in request전까지 timerid2를 시작하고, timerid1을 중단하는 기능을 담당.

Local Variables

**a2** : document.getElementbyid("logo”).innerText로 {{message}}

Return : #name h3, #stopsound, #start1, #stop2

function **mmm7** (<https://babycamip/?action=snapshot>)

감정인식을 위한 azure api의 json request 값을 context에 전달하는 javascript 함수로 5초 간격으로 값을 업데이트하여, 아이의 감정을 template에 수시로 띄워줌.

Local Variables

**Happiness, anger, neutral** : azure api 값에 대한 context를 정함.

Return : #name2 h1

function **playsound** (<div id ='sound'></div>)

안전 구역 경계 이탈 시에 알림음을 넣어주는 기능을 담당하는 함수로, 이탈 신호를 0.3초 간격으로 체크하여 이탈 시 지정된 알림음을 자동, 반복 재생함.

Local Variables

Return : <audio autoplay ="autoplay” loop src ="/static/music/siren.mp3”>

1. **Context**

{{message}}

{{babystate}}

{{hostip}}

{{maincamip}}

{{babycamip}}

1. **Request**

<https://maincamip/?action=stream>

<https://babycamip/?action=stream>

**baby.html**

경로 : Smart\_Kids\_Car\finalserver3\finalserver3\mysite\elections\templates\elections

Django에서 사용하는 template file로 view에서 받는 request 중에서 https://hostip/?action=main을 통해 받아오며, 사용자가 baby cam을 통해 차량에 탑승한 아동의 얼굴을 지켜보고, 감정상태를 체크하며, 사진을 찍는 기능 탑재됨.

1. **Dependencies:**

j-quary, javascript, html2canvas, requests, bootstrap, **static**

1. **Global Variables**

**timerid1** : 경계구역 이탈 여부를 0.3초 간격으로 체크하는 역할과 알림기능 담당.

**timerid2** : 안전구역 자동 복구 기능의 정상 작동 여부를 0.3초 간격으로 체크함.

**picture** : babycam으로부터 스트리밍한 아이의 영상을 받아오는 div임.

**capture** : babycam의 영상을 캡처한 순간의 frame을 업로드 하는 div임.

1. **Functions**

function **capture** (<div id="capture"></div>)

timerid1에 대한 javascript 함수로 0.3초 간격으로 message context를 업데이트하여 out request시에 알림음과 메세지 그리고 자동 페이지 넘김 설정 기능을 담당.

Local Variables

**a3** : document.getElementbyid("babycamip”).innerText를 저장.

Return : #save, <img src = http://babycamip/?action=snapshot/>

1. **Context**

{{message}}

{{babycamip}}

1. **Request**

<https://babycamip/?action=stream>