

# 차세대 분산 시스템 최종 보고서

## - 적외선 센서를 이용한 침입자 탐지 시스템 -

Project Team T2  
201311265 김상원  
201311269 김제헌

Date  
2017-12-07

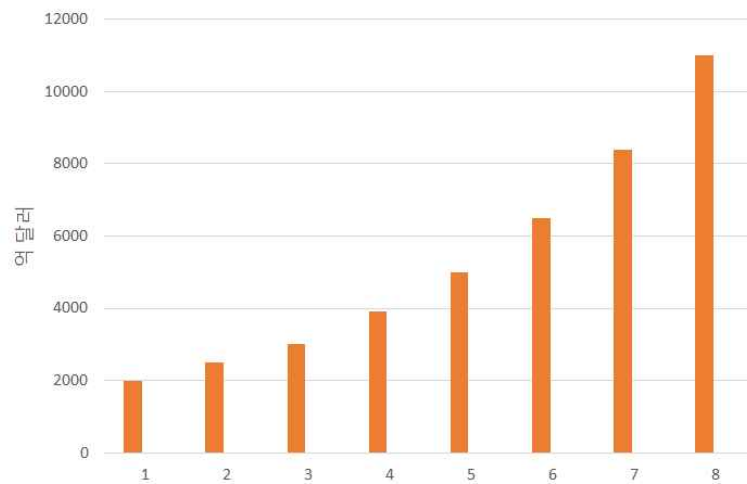
## 목차

1. 개발 동기 .....	3
2. 프로젝트 목표 .....	5
3. 초기 프로젝트 소개 .....	5
4. 변경된 프로젝트 소개 .....	7
5. 프로젝트 기능 .....	8
6. 프로젝트 실행 화면 .....	11
7. 구현 특이 사항 .....	13
8. 추후 개선 사항 .....	14

## 1. 개발 동기

### 1.1 점점 커지고 있는 세계 IoT 시장 전망

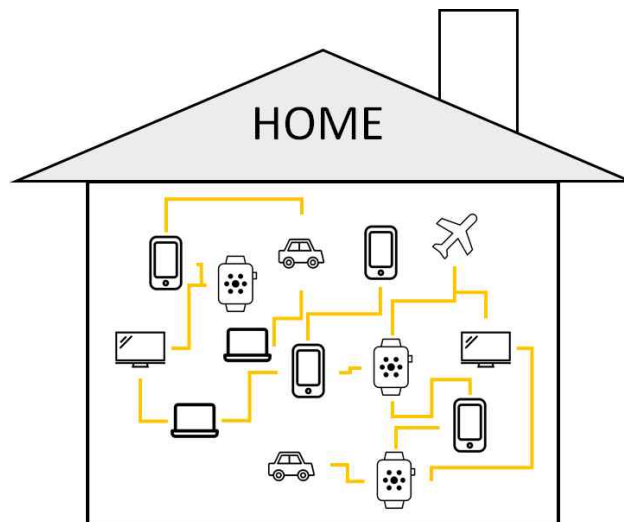
다음 [그림 1]은 세계 IoT의 시장 전망이 점점 더 커지고 있다는 그래프입니다. 세계적으로 IoT시장이 크게 성장하고 있는 와중에, 저희팀도 어떤 식으로 IoT가 이루어지고 개발되는지 한번 경험해보고자 해서 주제를 IoT로 정해보았습니다.



[그림 1]. 세계 IoT의 시장 전망

### 1.2 Home IoT

다음 [그림 2]는 홈 IoT를 의미하는 그림입니다. 그래서 IoT에 대해 알아보다가, IoT 중에서도 'IoT' 와 '스마트홈'을 결합한 '홈IoT'라는 이 있다는 것을 알게 되었습니다. 참고로 홈IoT란 스마트홈과 사물인터넷(IoT)을 결합한 단어입니다. 모바일 기기, 가전 등을 인터넷과 통신으로 모두 연결하여 정보를 수집하고 교환하는 플랫폼을 의미합니다. 이를 통해 집 밖에서도 스마트 기기를 이용하여 집 안의 가전제품을 제어하고 통제할 수 있습니다.



[그림 2] 홈 IoT

다음 [그림 3]은 현재 LG유플러스에서 제공해주고 있는 IoT@home 서비스에 대한 소개입니다.

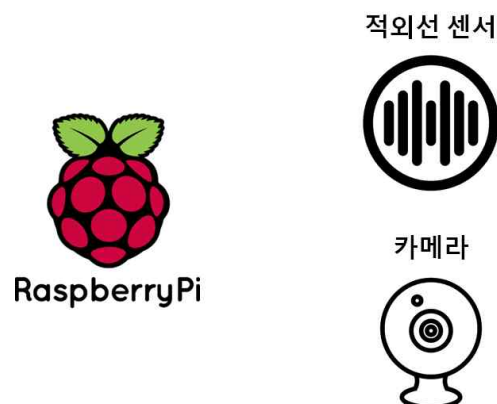
여기에는 열림감지센서, 가스락, 온도조절기, 도어락 등 다양한 부분을 원격으로 통제할 수 있도록 지원해 주고 있습니다. 제가 예전에 살던 집에서 쓰던 LG IoT@Home의 제품 중 창문에 센서를 설치하여 창문이 열려있거나 닫혀있는 것을 탐지해 주는 것이 있었는데, 그 당시 이런 것을 보고 상당히 좋은 기술이라고 생각했었습니다. 그래서 그런 종류의 센서 관련 프로그램을 개발해보면 어떨까 하고 생각해보았습니다. 그래서 이런 쪽으로 개발을 해보고자 결정했습니다.



[그림 3] LG 유플러스의 IoT@home 서비스

### 1.3 개발 접근 용이성

그러한 센서 관련 프로그램 중에서, [그림 4]처럼 비교적 구하기 쉬운 라즈베리파이와 라즈베리파이의 적외선센서, 카메라모듈이 있었고, 이를 이용하여 근접을 탐지하고 접근한 상대를 카메라를 통해 실시간 스트리밍, 일정시간 녹화기능을 제공하는 프로그램을 개발하는 것이 좋을 것 같다고 생각되었습니다.



[그림 4] 라즈베리파이와 적외선 센서, 카메라

## 2. 프로젝트 목표

### 2.1 침입자 탐지 시스템의 의의

- 귀중품 또는 접근제한 구역에 설치하여 침입 시도 및 접근 시도를 탐지할 수 있습니다.
- 침입 상황을 녹화하여 경고성 효과와 후속조치를 위한 대비를 가능하게 합니다.

### 2.2 침입자 탐지 시스템의 기능적인 목표

- 적외선 센서를 통해 1m이내 접근 시 카메라를 통해 30초간의 영상을 저장할 수 있습니다.
- 접근이 탐지되면 푸시알림을 전송하여 실시간 스트리밍으로 카메라가 찍는 영상을 확인할 수 있습니다.
- 웹페이지를 통해 저장된 동영상목록들을 확인하여 저장된 영상을 다시 볼 수 있고 최신 50개의 영상을 저장하여 볼 수 있습니다.

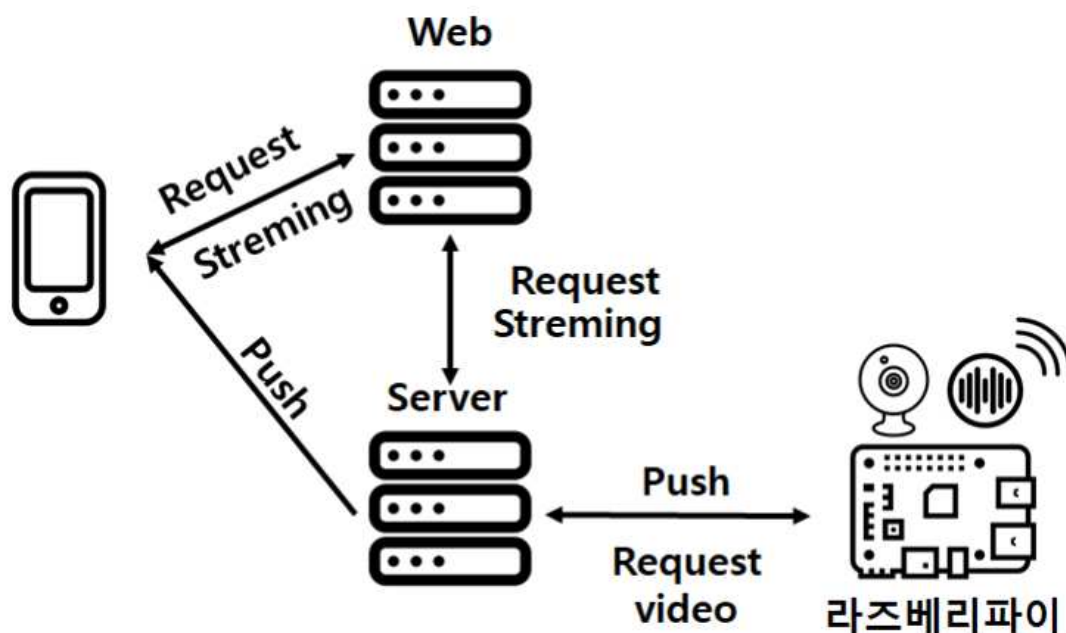
## 3. 초기 프로젝트 소개

### 3.1 프로젝트 제안서 발표 단계 때 초기에 구상했던 시스템 구조

전체적인 관점에서 IoT를 이용한 침입탐지 시스템은, 실질적으로 침입자를 감지하는 적외선 센서와, 동영상을 찍는 웹캠을 관리하는 라즈베리파이, 침입감지 및 영상 데이터를 받아서 전체를 관리하는 웹서버, 실질적인 동영상 확인을 할 수 있는 웹브라우저, 사용자에게 푸시알림을 전달해주기 위한 안드로이드로 구성되어있습니다.

그렇게 해서 나온 초기 프로젝트의 시스템 구조는 아래의 [그림 5]와 같습니다.

라즈베리파이가 관리하는 초음파 센서와 웹캠이 있고, 물체가 감지시, 이를 서버를 통해 휴대폰으로 전달해줍니다. 비디오는 웹으로 전달해주고, 휴대폰에서 이를 통해 동영상을 확인할 수 있다는 것을 표현한 그림입니다.



[그림 5] 초기 프로젝트 시스템 구조

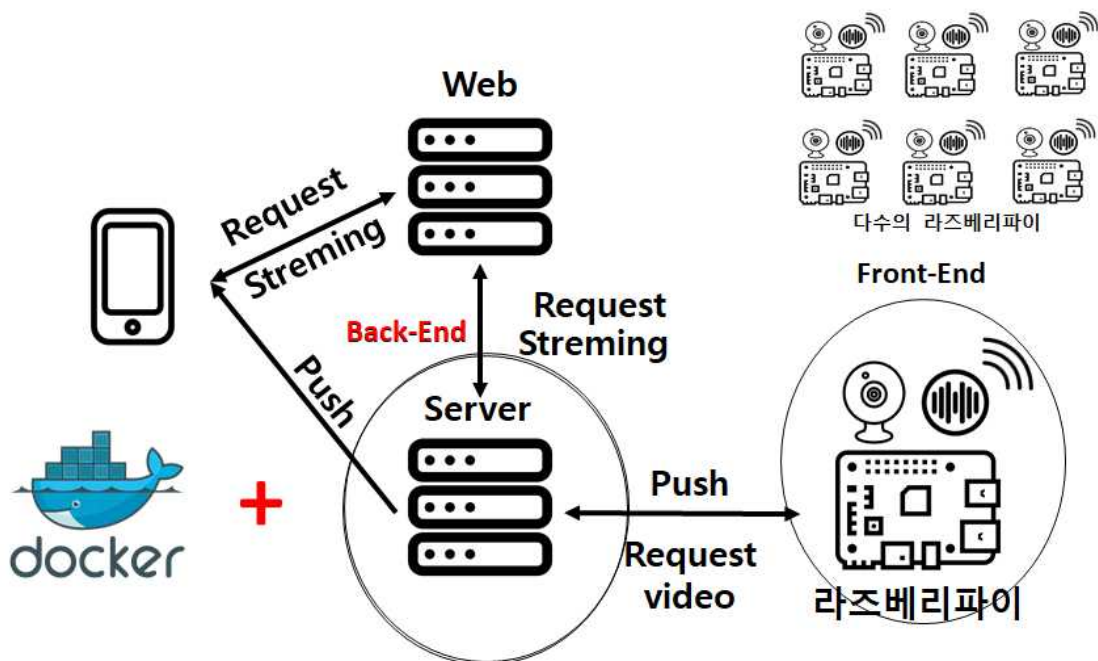
### 3.2 프로젝트가 변경된 이유

하지만 교수님께서 라즈베리파이, HTML, CSS, 아파치 웹서버, 파이썬, OpenCV, 안드로이드 등을 사용하고 있지만, 그저 이것저것 갖다 쓰는 것이고 '핵심이슈'를 다루고 있지 않고 있다고 지적해주셔서 바뀌게 되었습니다.

변경 가능했던 부분으로는 아래의 [그림 6]과 같이 적외선센서부분의 Front-End부분과 Server에서 이를 처리하는 Back-End부분이 있었습니다.

하지만 Front-End부분을 개선시키자니 라즈베리파이에서 적외선센서의 기능이 안 좋더라도 좋은 알고리즘을 통해 탐지 능력을 향상시키는 방안이 있었지만, 라즈베리파이부분은 잘 모르는 부분이며 인터넷에 있던 코드를 가져다 쓸 것이기 때문에 이를 향상시키는 것은 불가능할 것으로 판단되었고, Back-End를 향상시키는 방향으로 결정했습니다.

다수의 라즈베리파이가 있으면 메인서버 혼자서 이를 모두 처리할 때 부하량이 매우 크다고 판단되어 도커를 이용해서 이를 줄여보는 방향으로 결정하였습니다.



[그림 6] Back-End에 프로젝트의 중점을 두기로 결정함

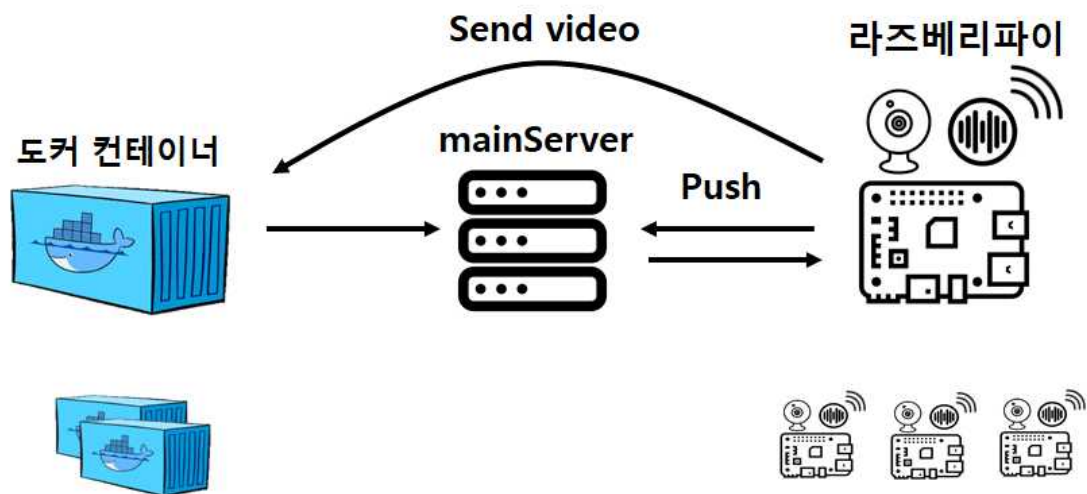
## 4. 변경된 프로젝트 소개

### 4.1 대략적인 시스템 구조

일단 저희가 프로젝트를 진행하면서 현실적인 벽에 많이 부딪혔습니다. 그중 가장 큰 것은 시간이라는 벽이었습니다. 이것저것 갖다 쓰더라도 어차피 새로 배우면서 공부해야 했던 것인데, 그러한 것들이 의미가 없다면, 시간도 부족하기 때문에 그러한 부가적인 것들을 빼버리고 핵심기능을 구현해보는 방향으로 정했습니다.

그래서 아래의 [그림 7]과 같이 기존의 라즈베리파이와 서버 외에 도커를 추가하였으며, 안드로이드, 웹을 빼게 되었습니다.

적외선센서가 감지 시, 라즈베리파이에서 도커 컨테이너로 동영상데이터를 전송하게 되며 도커 컨테이너에서 이를 받아서 저장을 하게 되며, 라즈베리파이의 탐지수가 많아지고 다시 적어짐에 따라 메인서버에서 도커 컨테이너의 수를 조정하는 오토스케일링 기능도 구현하였습니다.



[그림 7] 변경된 프로젝트 시스템 구조

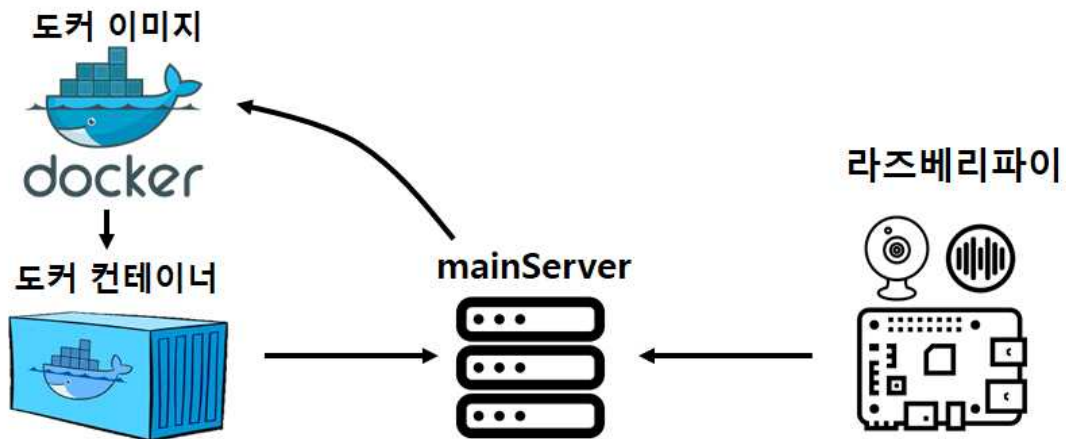
### 4.2 프로젝트 개발 환경

- 개발 환경 : Linux Mint 18.2 Sonya, Python3.5, OpenCV  
\*RaspberryPi = 카메라 모듈 고장
- 스트리밍 데이터 수신 에이전트 : Docker
- Testing 환경 : 노트북 웹캠 활용

## 5. 프로젝트 기능

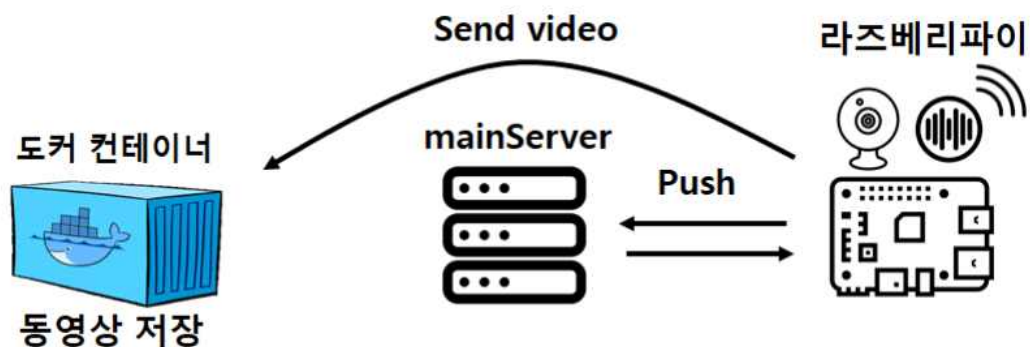
프로젝트의 실행 흐름 및 오토 스케일링 적용 설명

- 1. 일단 메인서버가 켜지면, 기존에 완성된 도커 이미지를 통해 도커 컨테이너를 생성합니다. 도커 컨테이너는 생성되자마자 메인서버에 접속합니다. 라즈베리파이도 켜지자마자 메인서버에 접속합니다. 메인서버는 이러한 목록을 가지고 있으며 이들을 조율하게 됩니다.



[그림 8] 메인서버가 켜진 이후의 실행 흐름

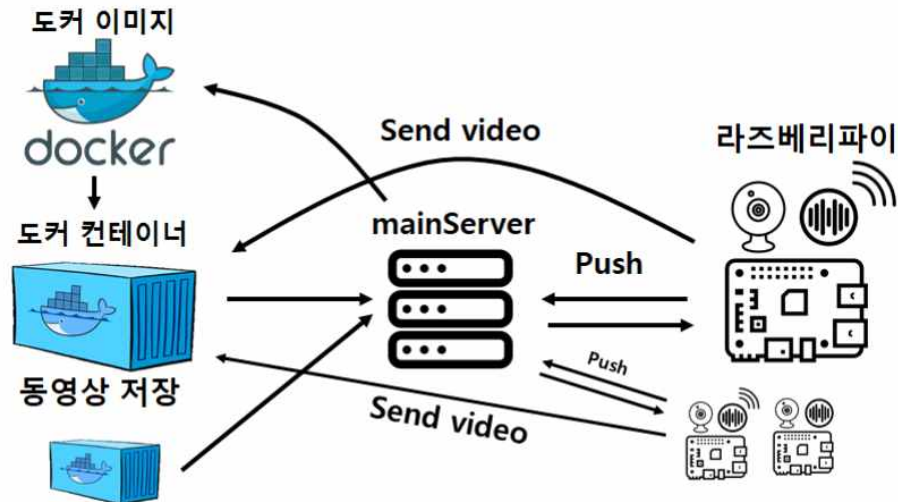
- 2. 라즈베리파이가 물체를 탐지시, 메인서버로 [Detected] 신호를 보내게되며, 메인서버는 이를 받고 라즈베리파이에게 연결할수 있는 도커 컨테이너의 주소(IP, PORT)를 넘겨주게 됩니다. 그러면 라즈베리파이는 이를 받아서 도커컨테이너에 접속을 하게 되며, UDP통신 및 OpenCV기술을 써서 도커 컨테이너로 동영상 전송하게 됩니다. 도커 컨테이너는 이를 받아서 호스트 운영체제인 mainServer쪽에 동영상을 저장하게 됩니다.



[그림 9] 라즈베리파이에서 물체가 탐지시 이후의 실행 흐름

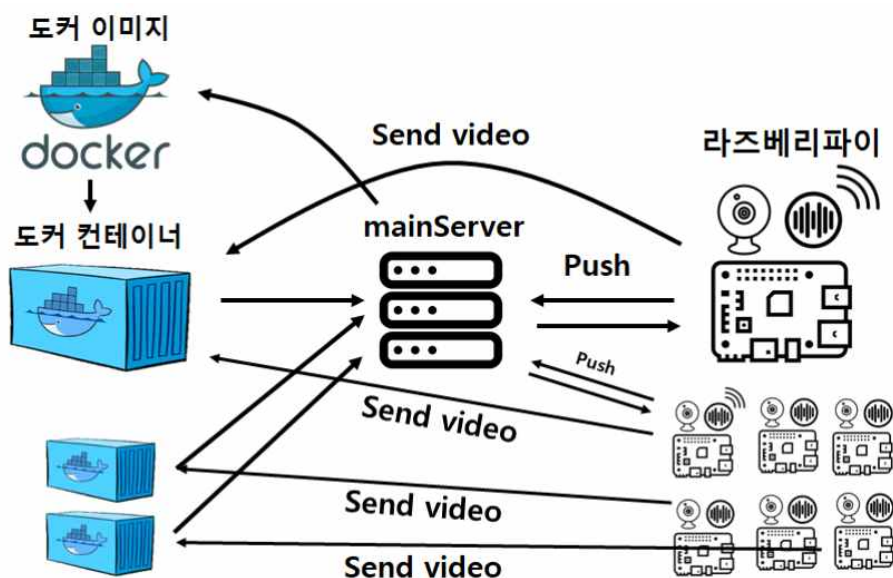


- 3. 이와 유사한 방식으로 라즈베리파이2, 3이 각각 메인서버에 접속해 있다가 물체를 탐지 시, 도커 컨테이너1에 접속을 하며, UDP 통신 및 OpenCV기술을 써서 동영상을 전송하게 됩니다. 이때 도커 컨테이너에서 자신과 연결되어 동영상을 전송하고 있는 라즈베리파이의 수가 3개가 될 경우, 서버로 [Full] 신호를 보내게 되며, 서버는 이를 받고 도커 이미지를 통해 새로운 도커 컨테이너2를 만들게 됩니다.



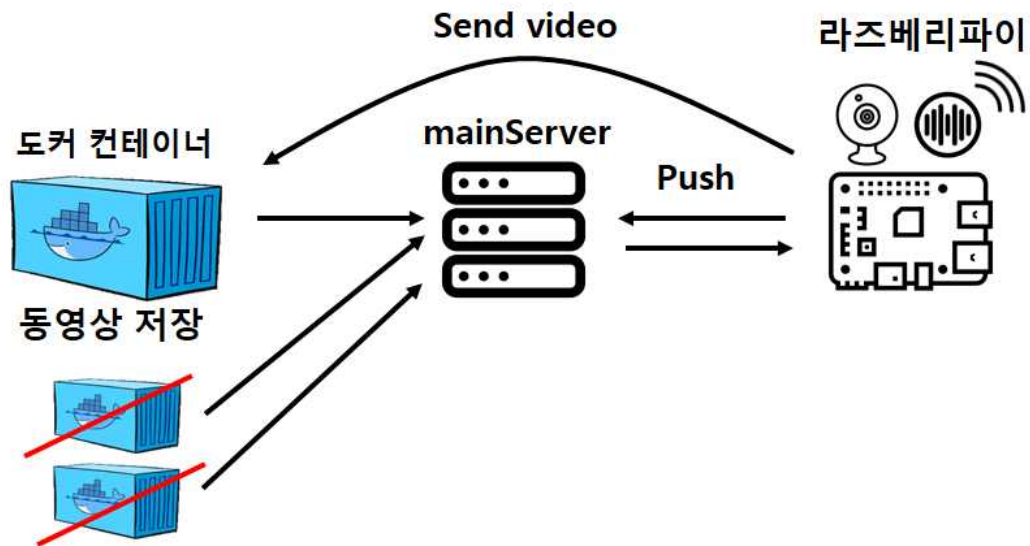
[그림 10] 추가적인 라즈베리파이2,3에서 물체가 탐지시 이후의 실행 흐름

- 4. 유사한 방식으로 라즈베리파이 4, 5, 6이 각각 메인서버에 접속해 있다가 물체를 탐지 시, 도커 컨테이너2에 접속을 하며, 동영상을 전송하고 도커는 이를 받아서 동영상을 저장합니다. 도커 컨테이너2에서도 자신과 연결되어 동영상을 전송중인 라즈베리파이의 수가 3개가 될 경우, 서버로 [Full]신호를 보내게 되며, 서버는 이를 받아서 도커 이미지를 통해 새로운 도커 컨테이너3을 만들게 됩니다. 이후 라즈베리파이 7에서 물체가 탐지 시 도커 컨테이너3으로 접속을 하게 됩니다.



[그림 11] 추가적인 라즈베리파이 4~7에서 물체가 탐지시 이후의 실행 흐름

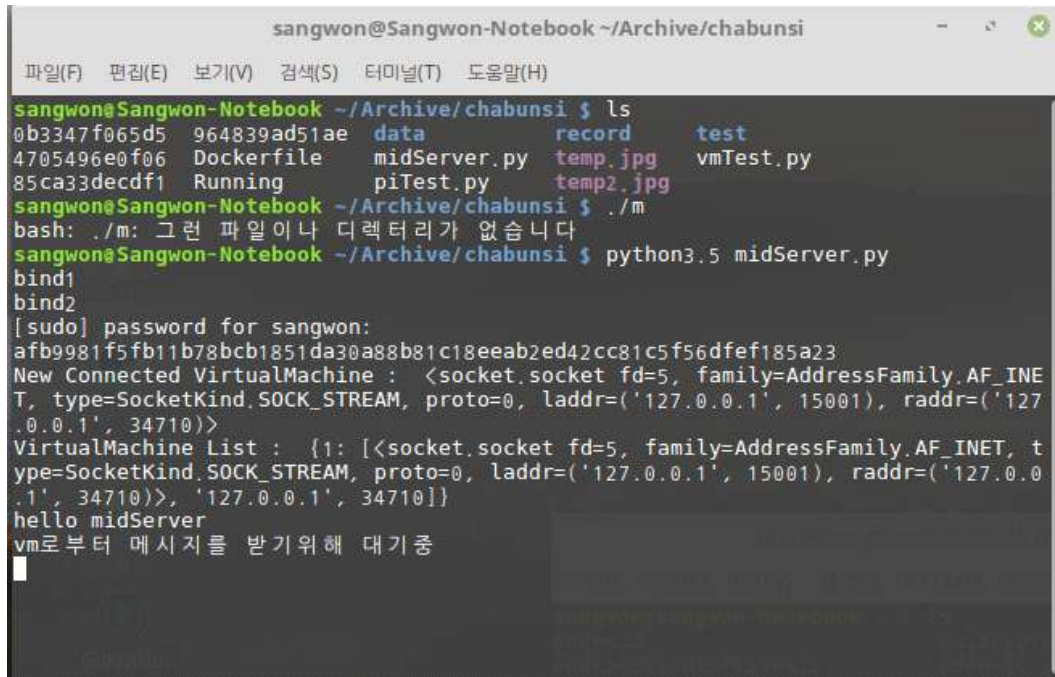
- 5. 라즈베리파이 2~7에서 동영상 전송을 마치게 될 경우 도커 컨테이너로 [Quit] 신호를 주게 되며, 도커 컨테이너에서는 이를 통해 자신과 연결된 라즈베리파이와의 연결을 종료하게 됩니다. 도커 컨테이너는 자신과 연결된 라즈베리파이의 수가 0이 될 경우, 메인서버로 [Empty] 신호를 보내게 되며 메인서버에서는 이를 받아서 최소 가동하는 컨테이너의 수를 1개만 놔둘 수 있게 하고 그 외의 컨테이너는 제거하게 됩니다.



[그림 11] 라즈베리파이 3~7에서 동영상 전송을 마침에 따라 이후의 실행 흐름

## 6. 프로젝트 실행 화면

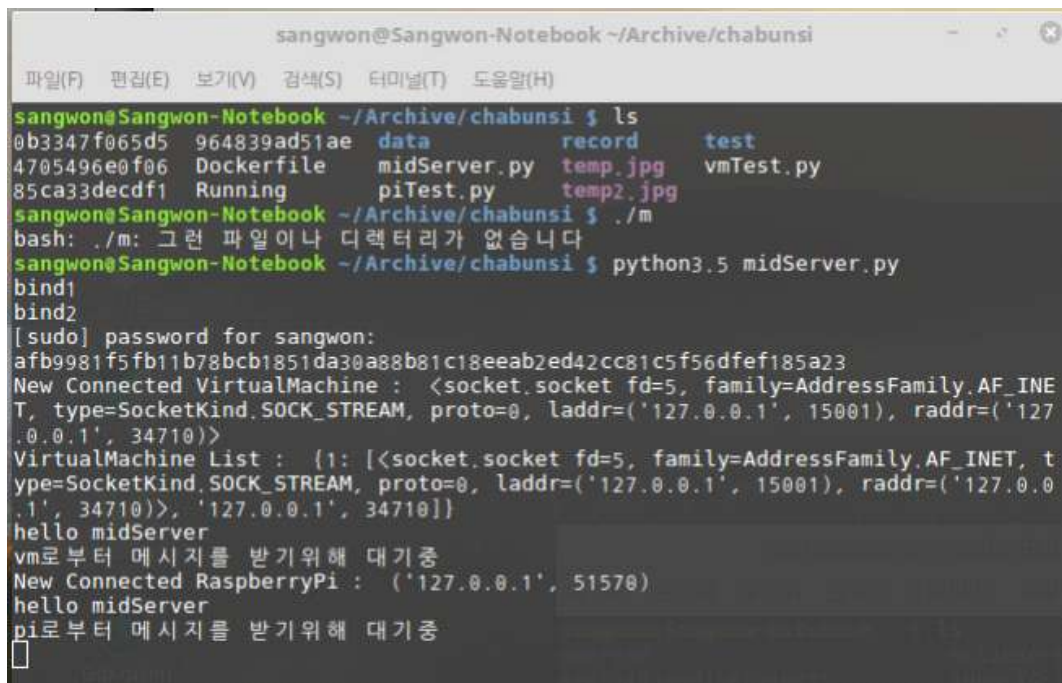
- 메인 서버의 실행 화면 : 메인 서버가 처음 시작되면 라즈베리 파이들의 접속을 받을 포트를 열고, 새로운 에이전트 컨테이너를 생성합니다. 컨테이너는 정상적으로 실행된 후 메인 서버에 접속하고 서버는 모니터링 스레드를 실행하여 부하정보를 모니터링 합니다.



```
sangwon@Sangwon-Notebook ~/Archive/chabunsi
sangwon@Sangwon-Notebook ~/Archive/chabunsi $ ls
0b3347f065d5 964839ad51ae data record test
4705496e0f06 Dockerfile midServer.py temp.jpg vmTest.py
85ca33decdf1 Running piTest.py temp2.jpg
sangwon@Sangwon-Notebook ~/Archive/chabunsi $ ./m
bash: ./m: 그런 파일이나 디렉터리가 없습니다
sangwon@Sangwon-Notebook ~/Archive/chabunsi $ python3.5 midServer.py
bind1
bind2
[sudo] password for sangwon:
afbf9981f5fb11b78bcb1851da30a88b81c18eeab2ed42cc81c5f56dfef185a23
New Connected VirtualMachine : <socket.socket fd=5, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 15001), raddr=('127.0.0.1', 34710)>
VirtualMachine List : {1: [<socket.socket fd=5, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 15001), raddr=('127.0.0.1', 34710)>, '127.0.0.1', 34710]}
hello midServer
vm로부터 메시지를 받기 위해 대기중
```

[그림 12] 메인 서버의 실행 화면

- 라즈베리 파이 하나가 작동한 후 메인서버에 접속한 상태

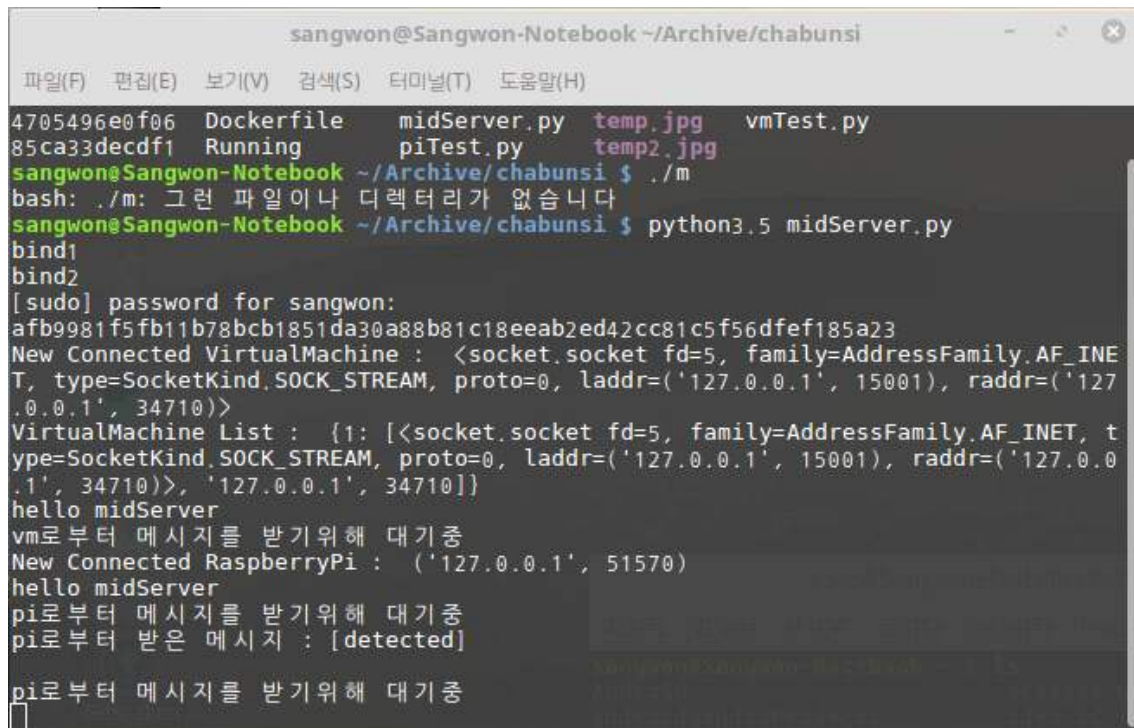


```
sangwon@Sangwon-Notebook ~/Archive/chabunsi
sangwon@Sangwon-Notebook ~/Archive/chabunsi $ ls
0b3347f065d5 964839ad51ae data record test
4705496e0f06 Dockerfile midServer.py temp.jpg vmTest.py
85ca33decdf1 Running piTest.py temp2.jpg
sangwon@Sangwon-Notebook ~/Archive/chabunsi $ ./m
bash: ./m: 그런 파일이나 디렉터리가 없습니다
sangwon@Sangwon-Notebook ~/Archive/chabunsi $ python3.5 midServer.py
bind1
bind2
[sudo] password for sangwon:
afbf9981f5fb11b78bcb1851da30a88b81c18eeab2ed42cc81c5f56dfef185a23
New Connected VirtualMachine : <socket.socket fd=5, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 15001), raddr=('127.0.0.1', 34710)>
VirtualMachine List : {1: [<socket.socket fd=5, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 15001), raddr=('127.0.0.1', 34710)>, '127.0.0.1', 34710]}
hello midServer
vm로부터 메시지를 받기 위해 대기중
New Connected RaspberryPi : ('127.0.0.1', 51570)
hello midServer
pi로부터 메시지를 받기 위해 대기중
```

[그림 13] 라즈베리 파이가 작동한 직후의 메인 서버의 실행화면

- 접속한 라즈베리 파이가 접근을 탐지한 후 영상을 전송

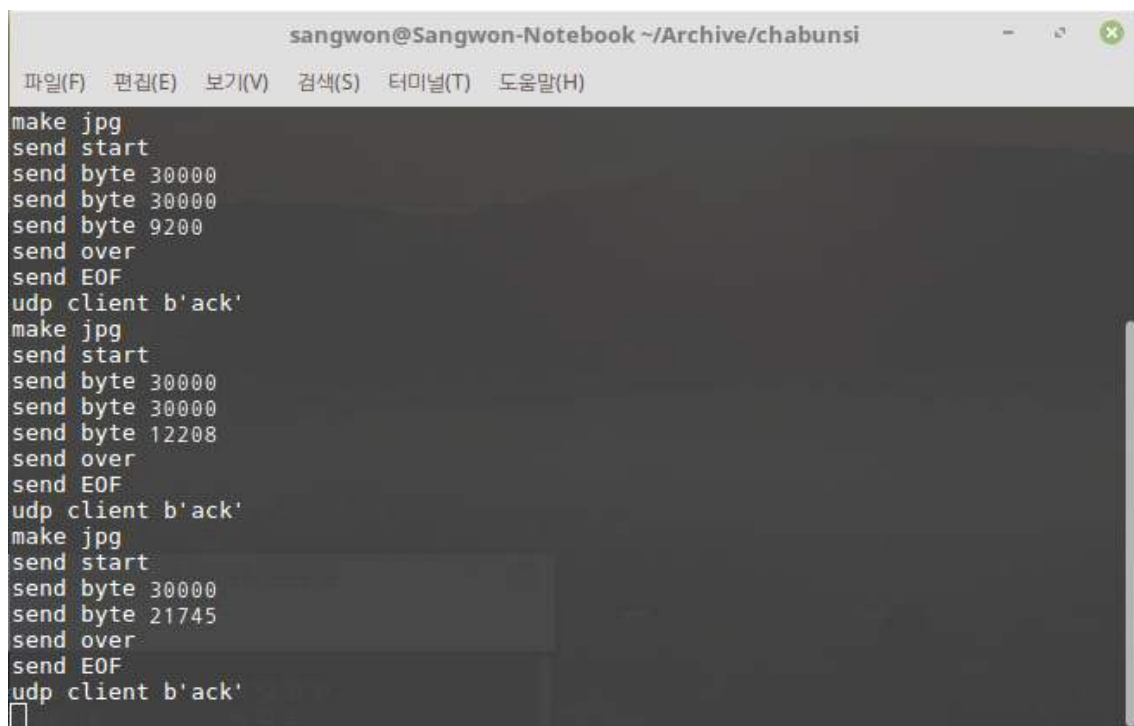
<메인 서버>



```
sangwon@Sangwon-Notebook ~/Archive/chabunsi
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
4705496e0f06 Dockerfile midServer.py temp.jpg vmTest.py
85ca33decdf1 Running piTest.py temp2.jpg
sangwon@Sangwon-Notebook ~/Archive/chabunsi $ ./m
bash: ./m: 그런 파일이나 디렉터리가 없습니다
sangwon@Sangwon-Notebook ~/Archive/chabunsi $ python3.5 midServer.py
bind1
bind2
[sudo] password for sangwon:
afb9981f5fb11b78bcb1851da30a88b81c18eeab2ed42cc81c5f56dfef185a23
New Connected VirtualMachine : <socket.socket fd=5, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 15001), raddr=('127.0.0.1', 34710)>
VirtualMachine List : {1: [<socket.socket fd=5, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 15001), raddr=('127.0.0.1', 34710)>, '127.0.0.1', 34710]}
hello midServer
vm로부터 메시지를 받기 위해 대기중
New Connected RaspberryPi : ('127.0.0.1', 51570)
hello midServer
pi로부터 메시지를 받기 위해 대기중
pi로부터 받은 메시지 : [detected]
pi로부터 메시지를 받기 위해 대기중
```

[그림 14] 메인 서버의 실행화면

<라즈베리 파이>

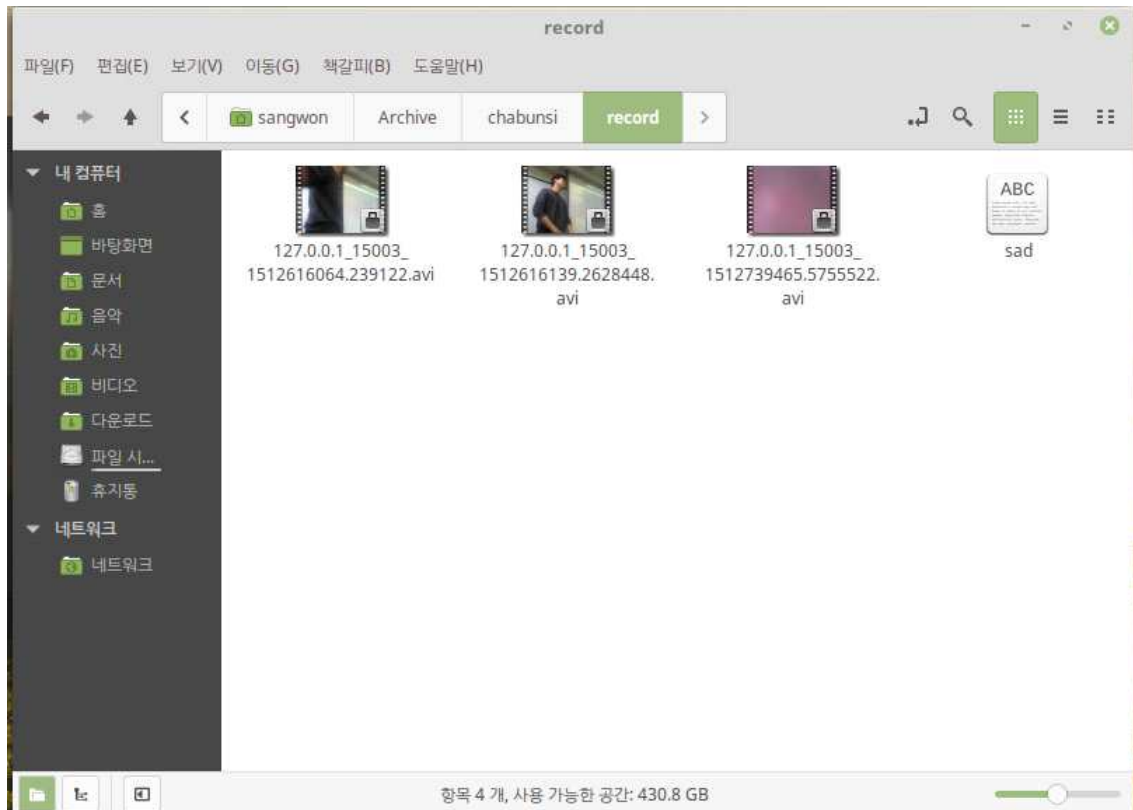


```
sangwon@Sangwon-Notebook ~/Archive/chabunsi
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
make jpg
send start
send byte 30000
send byte 30000
send byte 9200
send over
send EOF
udp client b'ack'
make jpg
send start
send byte 30000
send byte 30000
send byte 12208
send over
send EOF
udp client b'ack'
make jpg
send start
send byte 30000
send byte 21745
send over
send EOF
udp client b'ack'
```

[그림 15] 라즈베리 파이의 실행화면



- 라즈베리 파이가 전송한 영상은 컨테이너들의 공유 저장소에 저장됩니다.

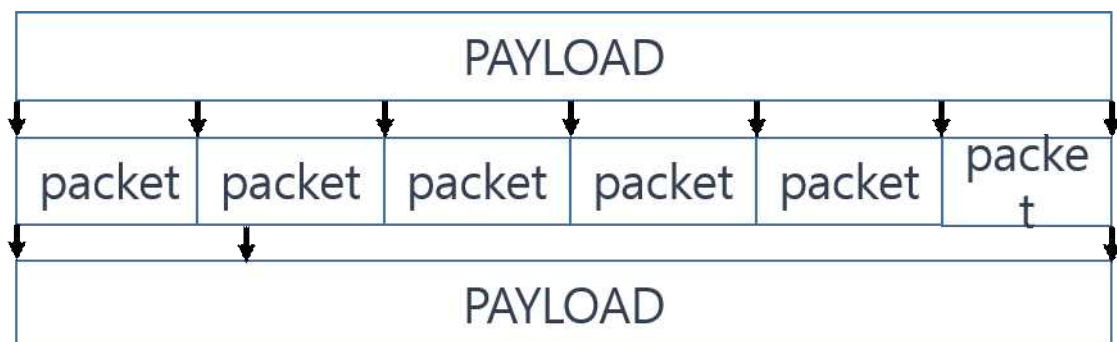


[그림 16] 각각의 도커 컨테이너에서 공유저장소에 저장한 동영상 목록

## 7. 구현 특이 사항

### 7.1 UDP 패킷 분할 전송

- UDP 패킷 Payload 크기 제한으로 인한 분할 전송 및 취합 기능



[그림 17] UDP 패킷 분할 전송

## 8. 추후 개선 사항

### 8.1 새로운 카메라 모듈

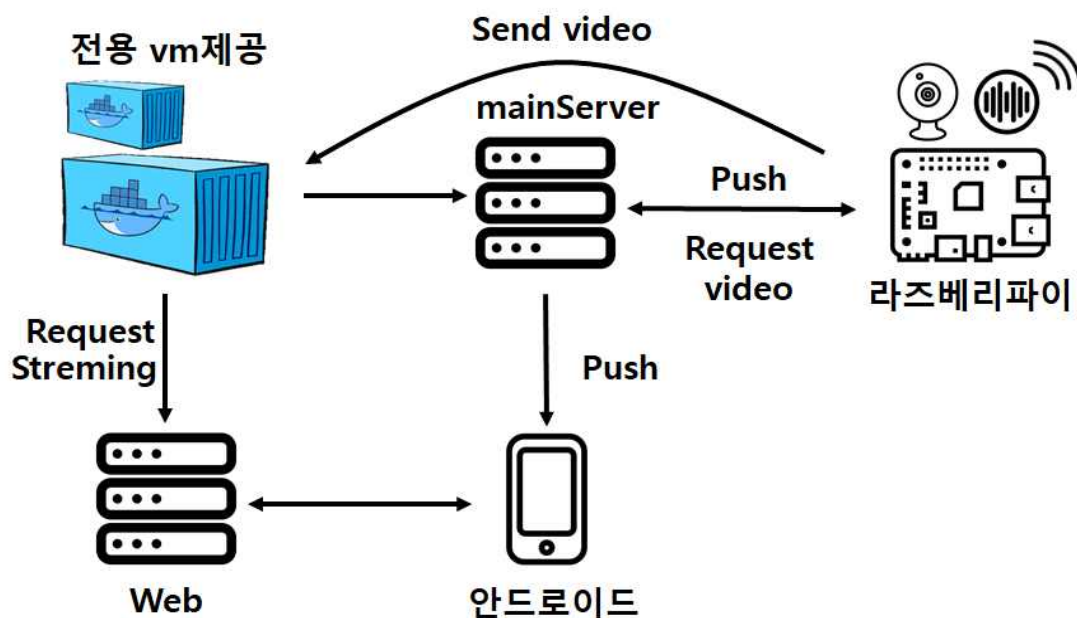
- 프로젝트를 진행하면서 라즈베리파이의 카메라 모듈이 망가져서 시연을 노트북 웹캠으로 밖에 할 수가 없었습니다. 그래서 해당 노트북에서는 하나의 라즈베리파이 역할밖에 할 수 없어 라즈베리파이 수가 늘어남에 따라 메인서버에서 VM의 수를 늘려주고 줄여주는 것을 보여줄 수가 없었습니다. 새로운 카메라 모듈을 구하게 되면 노트북 웹캠이 아닌, 좀 더 실제 IoT환경과 비슷한 환경에서 기능을 돌아가게 할 수가 있고, 오토스케일링 작업이 일어나는지도 확인할 수 있습니다. [오토스케일링 작업은 동영상인 아닌 텍스트 데이터를 보낼 때, 작동하는 것을 확인했습니다]

### 8.2 Web Server, Android Application

- 시간부족으로 인해 웹브라우저와 안드로이드 앱 연동을 배제한 채 구현하게 되었지만 이를 추가해서 구현하게 된다면, 웹브라우저를 통해 사용자의 서비스 접근성 향상과, 안드로이드의 푸시 알림 서비스를 통해 사용자 서비스 접근성 향상을 노릴 수가 있어 좀 더 실용적이게 됩니다.

### 8.3 차별화된 서비스 제공 가능

- 시간부족으로 포기하게 되었지만 저희가 맨 처음에 구상했던 모델로, 월 만원, 월 3만원 등의 각 모델에 따라, 최대 저장 가능한 영상의 수를 100개까지 지원해 주거나, 영상의 화질을 향상시켜준다든지, 전용VM[도커 컨테이너]을 지원해준다든지, 해당 집 내에서도 여러 라즈베리파이를 있을 경우 중요한 위치에 따라 중요도 값을 매겨 중요한 위치의 동영상을 먼저 보여주게 한다든지 등의 차별화된 서비스를 제공해 줄 수가 있고 이를 통한 수익성 모델을 구축할 수가 있습니다.



[그림 18] 추후 개선하면 좋을 것들