

홈네트워크

환자 감시 IoT 모니터링 시스템

2020261056 박지민

2020261069 한승현

2020261077 김찬규

2022261073 최윤아



목차

환자 감시 IOT 모니터링 시스템

1. 프로젝트 소개
2. 프로젝트 개요
3. 진행절차
4. 기능설명
5. 활용 기자재
6. 아쉬운 점

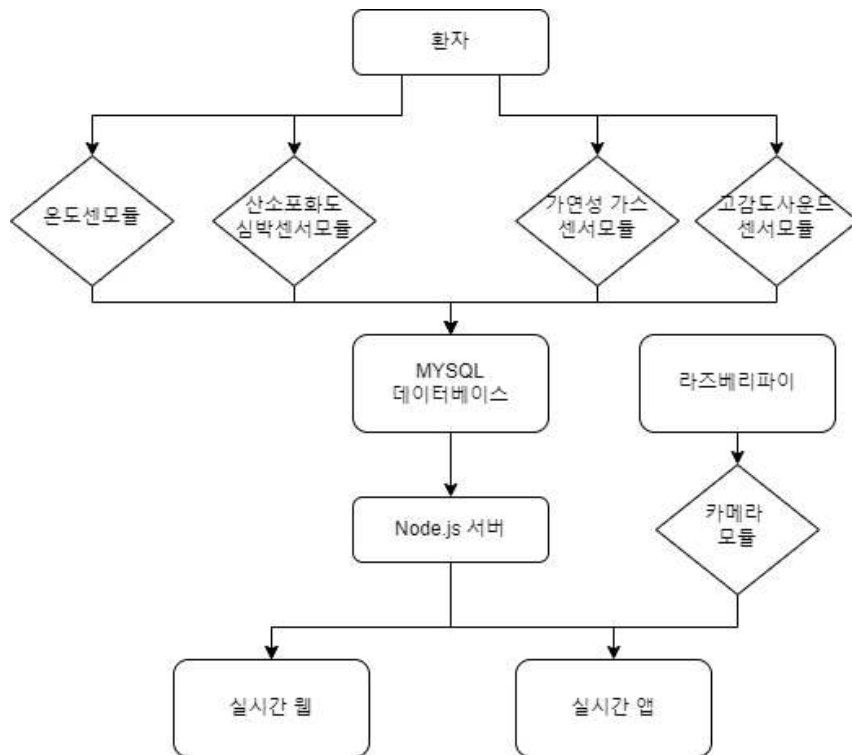
프로젝트 소개

“

환자의 보호자들이 언제 어디서나 환자의 건강상태를 확인할 수 있고 위급 상황 발생 또는 특정변화에 민감하게 반응하여 빠르게 확인할 수 있는 프로세스를 목적으로 하였습니다.

이를 위해 환자주변에서 센서를 사용해 데이터를 실시간으로 수집하고 저장하여 이용자에게 실시간으로 데이터를 제공하는 프로세스를 설계하였습니다.

프로젝트 개요



진행 절차

구분	기능	설명
센서값 측정	환자 정보 확인	정해진 주기(1초)에 따라 환자 정보 (심박수, 산소포화도, 온도, 소리, 가스) 실시간 측정
데이터베이스 구축	환자 정보 저장	환자 건강 상태 확인 및 이상 유무, 변화 등을 주기적 으로 확인하기 위해 라즈베리 파이에 데이터베이스 구축
데이터베이스 연동	환자 정보 연동	라즈베리 파이와 Window에 설치된 각각의 MySQL 데이터베이스를 연동함으로써 환자 정보 양방향 확인 가능

진행 절차

구분	기능	설명
데이터베이스 테이블 설계	환자 정보 저장	심박수, 산소포화도, 온도, 소리, 가스 데이터가 저장될 데이터베이스 테이블 설계
시리얼 통신	환자 정보 이동	아두이노에서 측정한 센서값을 정해진 주기(1초)에 따라 데이터베이스에 저장될 수 있도록 Python 시리얼 통신 설계
서버 구축	통신 중개	클라이언트와 데이터베이스 간의 통신 중개, 실시간 데이터 처리 및 전송을 위해 Node.js 서버 구축

진행 절차

구분	기능	설명
웹 화면 설계	환자 정보 확인	- 웹 화면에서 정해진 주기(3초)에 따라 데이터베이스에 마지막으로 저장된 환자 정보를 가져온 후 환자 건강 상태 실시간 확인
앱 화면 설계	환자 정보 확인	- React Native, Expo를 활용하여 앱 화면에서 정해진 주기(3초)에 따라 데이터베이스에 마지막으로 저장된 환자 정보를 바탕으로 환자 건강 상태 실시간 확인
웹캠 스트리밍	환자 상태 확인	- 라즈베리 파이 카메라를 통해 웹캠 스트리밍 구축, 웹에서 환자 상태 실시간 확인 가능

센서값 측정

```
#INCLUDE <WIRE.H>
#include "MAX30100_PULSEOXIMETER.H"
#define REPORTING_PERIOD_MS 1000 // 데이터를 보고하는 주기를 1초로 설정
PULSEOXIMETER POX; // PULSEOXIMETER 객체 생성
uint32_t TSLASTREPORT = 0; // 마지막으로 데이터를 보고한 시간을 저장하는 변수 선언

// LM35
float TEMPERATURE;
const int LM35PIN = A0;

// SOUND
int SOUND;
const int SOUNDPIN = A3;

// GAS
const int GASPIN = A6;
```


센서값 측정

```
VOID SETUP()
```

```
{
```

```
  SERIAL.BEGIN(9600);
```

```
  SERIAL.PRINTLN("산소포화도, 심박 센서 모듈을 초기화합니다.");
```

```
  // 센서 초기화에 실패하면 "실패"라는 메시지를 시리얼 모니터에 출력, 초기화에 성공하면 "성공"라는 메시지를  
  시리얼 모니터에 출력
```

```
  IF (!POX.BEGIN()) {
```

```
    SERIAL.PRINTLN("초기화 실패!");
```

```
    FOR(;;);
```

```
  } ELSE {
```

```
    SERIAL.PRINTLN("초기화 성공!");
```

```
  }
```

```
  // GAS
```

```
  PINMODE(GASPIN, INPUT);
```

```
}
```

센서값 측정

```
VOID LOOP()
```

```
{
```

```
    // LM35
```

```
    TEMPERATURE = ANALOGREAD(LM35PIN) * 0.48828125;
```

```
    // SOUND
```

```
    SOUND = ANALOGREAD(SOUNDPIN);
```

```
    POX.UPDATE(); // 센서 데이터 업데이트
```

```
    // 마지막으로 데이터를 보고한 시간과 현재 시간의 차이가 REPORTING_PERIOD_MS, 즉 1초보다 크면 아래 코드 수행
```

센서값 측정

```
IF (MILLIS() - TSLASTREPORT > REPORTING_PERIOD_MS) {
```

```
    FLOAT HEARTRATE = POX.GETHEARTRATE();
```

```
    FLOAT SP02 = POX.GETSP02();
```

```
    SERIAL.PRINT(HEARTRATE);
```

```
    SERIAL.PRINT(" , ");
```

```
    SERIAL.PRINT(SP02);
```

```
    SERIAL.PRINT(" , ");
```

```
    SERIAL.PRINT(TEMPERATURE);
```

```
    SERIAL.PRINT(" , ");
```

```
    SERIAL.PRINT(SOUND);
```

```
    SERIAL.PRINT(" , ");
```

```
    SERIAL.PRINTLN(ANALOGREAD(GASPIN));
```

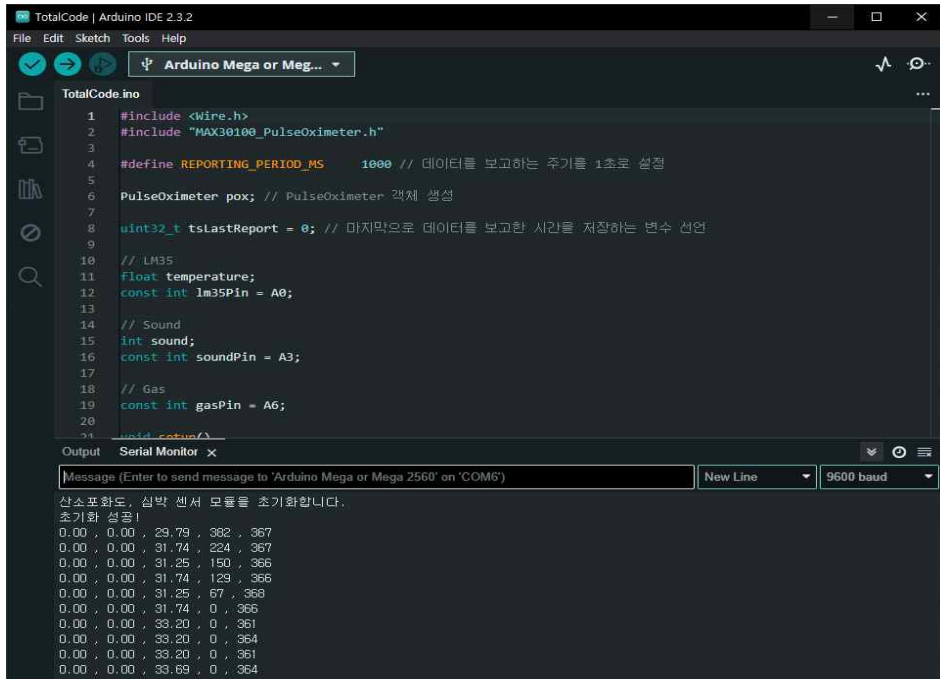
```
    TSLASTREPORT = MILLIS(); // 현재 시간을 TSLASTREPORT 변수에 저장
```

```
}
```

```
}
```

실행 결과

센서값 측정



The screenshot shows the Arduino IDE interface with the file 'TotalCode.ino' open. The code includes headers for Wire and MAX30100_PulseOximeter, defines a reporting period of 1000ms, and initializes variables for temperature, sound, and gas sensors. The serial monitor is open, displaying the output of the program.

```
TotalCode.ino
1 #include <Wire.h>
2 #include "MAX30100_PulseOximeter.h"
3
4 #define REPORTING_PERIOD_MS    1000 // 데이터를 보고하는 주기를 1초로 설정
5
6 PulseOximeter pox; // PulseOximeter 객체 생성
7
8 uint32_t tsLastReport = 0; // 마지막으로 데이터를 보고한 시간을 저장하는 변수 선언
9
10 // LM35
11 float temperature;
12 const int lm35Pin = A0;
13
14 // Sound
15 int sound;
16 const int soundPin = A3;
17
18 // Gas
19 const int gasPin = A6;
20
21 void setup()
```

Serial Monitor Output:

```
Message (Enter to send message to 'Arduino Mega or Mega 2560' on 'COM6')
산소포화도, 심박 센서 모듈을 초기화합니다.
초기화 성공!
0.00 , 0.00 , 29.79 , 382 , 367
0.00 , 0.00 , 31.74 , 224 , 367
0.00 , 0.00 , 31.25 , 150 , 366
0.00 , 0.00 , 31.74 , 129 , 366
0.00 , 0.00 , 31.25 , 67 , 368
0.00 , 0.00 , 31.74 , 0 , 366
0.00 , 0.00 , 33.20 , 0 , 361
0.00 , 0.00 , 33.20 , 0 , 364
0.00 , 0.00 , 33.20 , 0 , 361
0.00 , 0.00 , 33.69 , 0 , 364
```



데이터베이스 구축

패키지 목록 업데이트,
패키지 최신 버전 업그레이드

```
sudo apt-get update  
sudo apt-get upgrade
```

MySQL 설치

```
sudo apt-get install mysql-server mysql-client  
systemctl status mysql
```

데이터베이스 접속

```
sudo mysql -u root
```

데이터베이스 구축

root 계정 패스워드 변경

```
MariaDB (none)]> use mysql
```

```
// 패스워드 변경 전 계정 정보 확인
```

```
MariaDB mysql]> select user, host, password from user;
```

```
// 패스워드를 'password' 로 변경
```

```
MariaDB mysql]> UPDATE user SET password=PASSWORD('password' where user='root'
```

```
// 저장
```

```
MariaDBmysql]> FLUSH PRIVILEGES;
```

```
// 패스워드 변경 후 계정 정보 확인
```

```
MariaDB mysql]> select user, host, password from user;
```

```
// 종료
```

```
MariaDBmysql]> quit
```

데이터베이스 구축

외부접속 허용

```
pi@raspberrypi:~ $ cd /etc/mysql/mariadb.conf.d/
```

```
pi@raspberrypi:/etc/mysql/mariadb.conf.d/ $ ls
```

```
50-client.conf  50-mysql-clients.cnf  50-mysqld_safe.cnf  50-server.cnf
```

```
pi@raspberrypi:/etc/mysql/mariadb.conf.d/ $ sudo nano 50-server.cnf
```

(수정내용) bind-address = 127.0.0.1 -> bind-address = 0.0.0.0

```
pi@raspberrypi:~ $ sudo service mysql restart
```

데이터베이스 연동

Window MySQL 데이터베이스 연동

1) MySQL Workbench 실행

2) 상단 [Database]

-> [Manage Connections..] 선택

3) 하단 [New] 선택

4) 아래와 같이 입력

- Connection Name : raspberrypi

- Hostname : 192.168.45.82

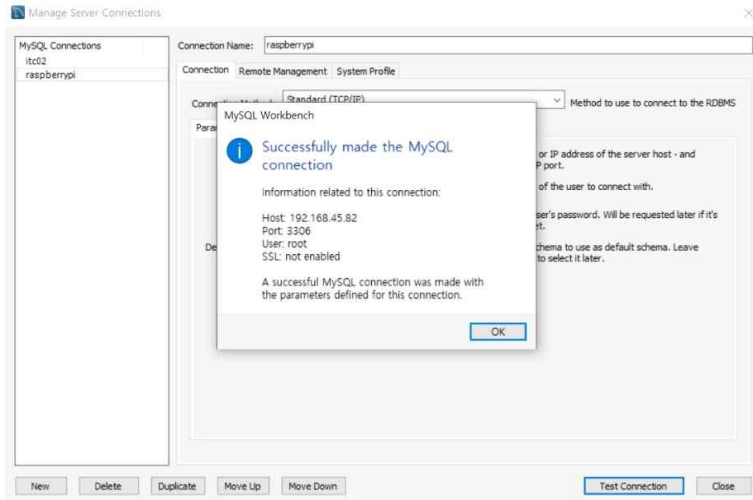
- Username : root

5) [password] -> [Store in Vault...] 선택

6) Password 입력 및 [OK] 선택

7) 하단 [Test Connection] 선택

8) [Successfully made the MySQL connection]
alert 창 확인



데이터베이스 테이블 설계

테이블 설계

USE project;

```
CREATE TABLE IF NOT EXISTS information (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    heart_rate FLOAT NOT NULL,  
    spo2 FLOAT NOT NULL,  
    temperature FLOAT NOT NULL,  
    sound INT NOT NULL,  
    gas INT NOT NULL,  
    measured_at TIMESTAMP DEFAULT
```

CURRENT_TIMESTAMP

);

Result Grid							
		Filter Rows:		Edit:		Exp	
id	heart_rate	spo2	temperature	sound	gas	measured_at	
NULL	NULL	NULL	NULL	NULL	NULL	NULL	

information(환자 정보)

PK	Column Name	Datatype	Description
O	id	INT(11)	아이디
	heart_rate	FLOAT	심박수
	spo2	FLOAT	산소포화도
	temperature	FLOAT	온도
	sound	INT(11)	소리
	gas	INT(11)	가스
	measured_at	TIMESTAMP	데이터 측정 시간

시리얼 통신

파이썬 가상환경 설정

```
> python -m venv project
```

pyserial,
pymysql 설치

```
> pip install pyserial
```

```
> pip install pymysql
```

시리얼 통신

```
import serial
import pymysql
from datetime import datetime
```

MySQL 연결 설정

```
db_config = {
    'user': 'root',
    'password': 'password', # 사용자 비밀번호로 변경
    'host': '192.168.45.82',
    'database': 'project'
}
```

아두이노와 시리얼 통신 설정

```
ser = serial.Serial('COM6', 9600) # 아두이노가 연결된 포트로 변경
```

데이터베이스 연결

```
db = pymysql.connect(**db_config)
cursor = db.cursor()
```

home.py

try:

while True:

시리얼 포트에서 데이터 읽기

line = ser.readline().decode('utf-8').strip()

data = line.split(',')

if len(data) == 5:

heart_rate = float(data[0])

spo2 = float(data[1])

temperature = float(data[2])

sound = int(data[3])

gas = int(data[3])

현재 시간 가져오기

current_time = datetime.now().strftime('%Y-%m-%d

%H:%M:%S')

데이터베이스에 값 삽입

insert_query = "INSERT INTO information

(heart_rate, spo2, temperature, sound, gas, measured_at)

VALUES (%s, %s, %s, %s, %s, %s)"

cursor.execute(insert_query, (heart_rate, spo2,
temperature, sound, gas, current_time))

db.commit()

print(f"Inserted heart_rate: {heart_rate}, spo2:
{spo2}, temperature: {temperature}, sound: {sound}, gas:
{gas} at {current_time}")

home.py

```
except KeyboardInterrupt:
```

```
    pass
```

```
finally:
```

```
    ser.close()
```

```
    cursor.close()
```

```
    db.close()
```

```
python home.py    #home.py 실행
```

Result Grid

Filter Rows:

Edit:

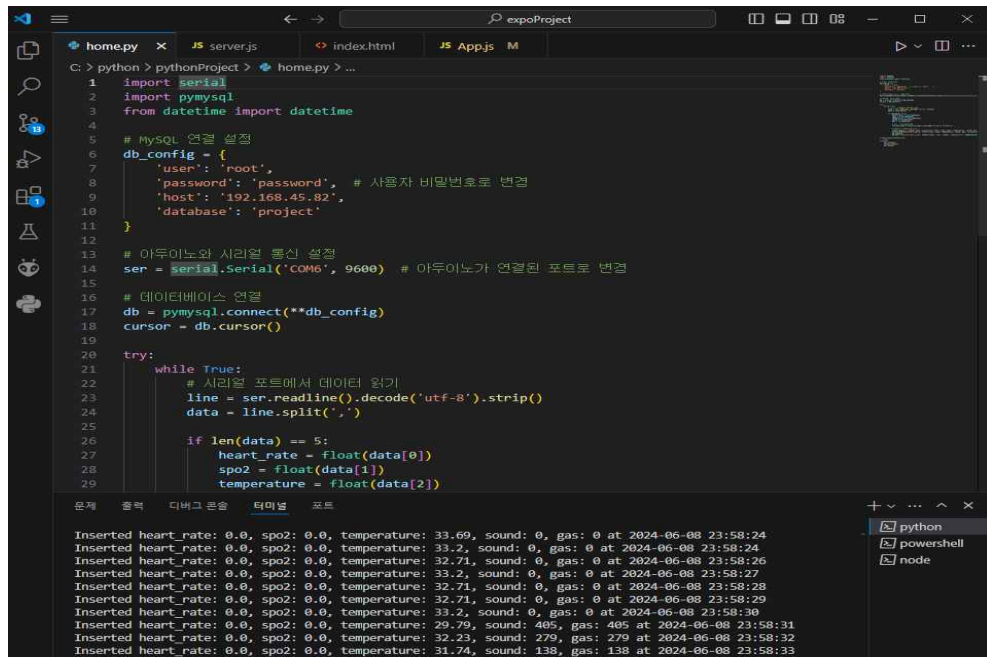
Export/Import

	id	heart_rate	spo2	temperature	sound	gas	measured_at
	7	0	0	31.25	337	326	2024-06-10 20:13:02
	8	0	0	30.76	177	330	2024-06-10 20:13:03
	9	21.88	0	30.76	148	328	2024-06-10 20:13:04
	10	21.88	0	30.76	114	334	2024-06-10 20:13:05
	11	42.12	0	30.76	0	327	2024-06-10 20:13:06
	12	42.12	0	30.76	0	330	2024-06-10 20:13:07
	13	43.79	0	32.71	0	325	2024-06-10 20:13:08
	14	51.96	95	32.71	0	326	2024-06-10 20:13:09
	15	88.88	95	30.76	0	329	2024-06-10 20:13:10
	16	271.78	94	31.25	0	326	2024-06-10 20:13:11

(데이터베이스로 데이터 전송 및 저장 확인)

실행 결과

시리얼 통신



```
1 import serial
2 import pymysql
3 from datetime import datetime
4
5 # MySQL 연결 설정
6 db_config = {
7     'user': 'root',
8     'password': 'password', # 사용자 비밀번호로 변경
9     'host': '192.168.45.82',
10    'database': 'project'
11 }
12
13 # 아두이노와 시리얼 통신 설정
14 ser = serial.Serial('COM6', 9600) # 아두이노가 연결된 포트로 변경
15
16 # 데이터베이스 연결
17 db = pymysql.connect(**db_config)
18 cursor = db.cursor()
19
20 try:
21     while True:
22         # 시리얼 포트에서 데이터 읽기
23         line = ser.readline().decode('utf-8').strip()
24         data = line.split(',')
25
26         if len(data) == 5:
27             heart_rate = float(data[0])
28             spo2 = float(data[1])
29             temperature = float(data[2])
30
31             # 데이터베이스에 데이터 저장
32             sql = "INSERT INTO sensor_data (heart_rate, spo2, temperature, sound, gas) VALUES (%s, %s, %s, %s, %s)"
33             cursor.execute(sql, (heart_rate, spo2, temperature, 0, 0))
34             db.commit()
35
36             # 로그 출력
37             print(f"Inserted heart_rate: {heart_rate}, spo2: {spo2}, temperature: {temperature}, sound: 0, gas: 0 at {datetime.now()}")
38
39 except KeyboardInterrupt:
40     print("Program interrupted by user.")
41 finally:
42     cursor.close()
43     db.close()
```

문제 출력 디버그 콘솔 터미널 포트

```
Inserted heart_rate: 0.0, spo2: 0.0, temperature: 33.69, sound: 0, gas: 0 at 2024-06-08 23:58:24
Inserted heart_rate: 0.0, spo2: 0.0, temperature: 33.2, sound: 0, gas: 0 at 2024-06-08 23:58:24
Inserted heart_rate: 0.0, spo2: 0.0, temperature: 32.71, sound: 0, gas: 0 at 2024-06-08 23:58:26
Inserted heart_rate: 0.0, spo2: 0.0, temperature: 33.2, sound: 0, gas: 0 at 2024-06-08 23:58:27
Inserted heart_rate: 0.0, spo2: 0.0, temperature: 32.71, sound: 0, gas: 0 at 2024-06-08 23:58:28
Inserted heart_rate: 0.0, spo2: 0.0, temperature: 32.71, sound: 0, gas: 0 at 2024-06-08 23:58:29
Inserted heart_rate: 0.0, spo2: 0.0, temperature: 33.2, sound: 0, gas: 0 at 2024-06-08 23:58:30
Inserted heart_rate: 0.0, spo2: 0.0, temperature: 29.79, sound: 405, gas: 405 at 2024-06-08 23:58:31
Inserted heart_rate: 0.0, spo2: 0.0, temperature: 32.23, sound: 279, gas: 279 at 2024-06-08 23:58:32
Inserted heart_rate: 0.0, spo2: 0.0, temperature: 31.74, sound: 138, gas: 138 at 2024-06-08 23:58:33
```

서버 구축

Node.js 프로젝트 초기화 및
package.json 파일 생성

```
npm init -y
```

Express, MySQL,
Socket.IO 패키지 설치

```
npm install express mysql socket.io
```

server.js

```
const express = require('express');
const mysql = require('mysql');
const http = require('http');
const socketio = require('socket.io');
// MySQL 연결 설정
const db = mysql.createConnection({
  host: '192.168.45.82',
  user: 'root',
  password: 'password',
  database: 'project'
});
```

```
// MySQL 연결 시도
```

```
db.connect((err) => {
```

```
  if (err) {
```

```
    throw err;
```

```
  }
```

```
  console.log('MySQL 데이터베이스 연결 성공');
```

```
});
```

```
const app = express(); // Express 애플리케이션 인스턴스 생성
```

```
const server = http.createServer(app); // HTTP 서버 생성, Express 앱 사용
```

```
const io = socketio(server); // Socket.IO 서버 생성, HTTP 서버 사용
```

```
app.use(express.static('public')); // public 디렉터리 내 정적 파일(CSS 등)을  
제공하도록 설정
```


server.js

// 루트 경로('/')에 대한 GET 요청을 처리하는 핸들러 정의

// 클라이언트에게 "실시간 서버에 오신 것을 환영합니다."
응답 전송

```
app.get('/', (req, res) => {  
  res.send('실시간 서버에 오신 것을 환영합니다.');
```

```
});  
// 새로운 클라이언트가 연결되었을 때 실행되는 콜백 함수  
정의
```

```
io.on('connection', (socket) => {  
  console.log('새로운 클라이언트가 연결되었습니다.');
```

// 실시간 데이터 전송

```
setInterval(() => {  
  db.query('SELECT * FROM information ORDER BY measured_at  
DESC LIMIT 1', (err, result) => {  
    if (err) throw err;  
    socket.emit('FromAPI', result[0]);  
  });  
}, 3000);
```

// 클라이언트 연결이 끊어졌을 때 실행되는 콜백 함수 정의

```
socket.on('disconnect', () => {  
  console.log('클라이언트 연결이 중단되었습니다.');
```

```
});  
const port = 3000; // 포트 번호 설정
```

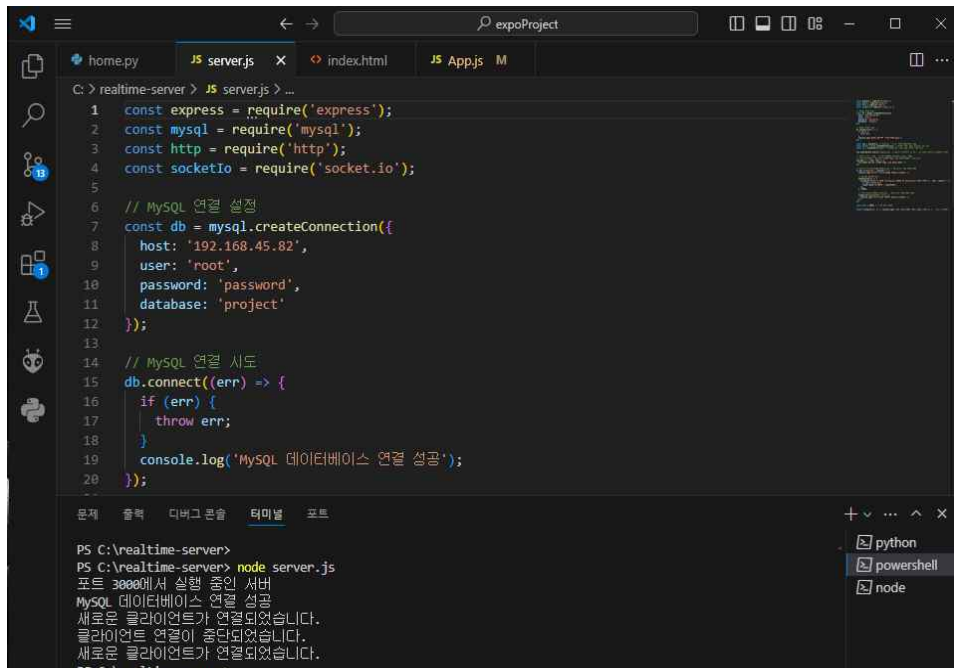
```
server.listen(port, () => console.log(` 포트 ${port}에서 실행 중인  
서버`)); // 지정된 포트에서 서버 시작, 콘솔 메시지 출력
```

server.js 실행

node server.js

실행 결과

서버 실행



The screenshot shows a Visual Studio Code editor window with a file named `server.js` open. The code defines an Express server with MySQL and Socket.io integration. The terminal at the bottom shows the command `node server.js` being executed, with output indicating successful database connection and client connections.

```
C:\> realtime-server > JS server.js > ...
1  const express = require('express');
2  const mysql = require('mysql');
3  const http = require('http');
4  const socketIo = require('socket.io');
5
6  // MySQL 연결 설정
7  const db = mysql.createConnection({
8    host: '192.168.45.82',
9    user: 'root',
10   password: 'password',
11   database: 'project'
12 });
13
14 // MySQL 연결 시도
15 db.connect((err) => {
16   if (err) {
17     throw err;
18   }
19   console.log('MySQL 데이터베이스 연결 성공');
20 });
```

문제 출력 디버그 콘솔 터미널 포트

```
PS C:\realtime-server>
PS C:\realtime-server> node server.js
포트 3000에서 실행 중인 서버
MySQL 데이터베이스 연결 성공
새로운 클라이언트가 연결되었습니다.
클라이언트 연결이 중단되었습니다.
새로운 클라이언트가 연결되었습니다.
```

웹 설계

Index.html

```
<!DOCTYPE html>
<html>
<head>
<title>Vital-Track</title>
<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">
<style>
body {
background-image: url('https://preview.free3d.com/img/2017/07/2398830626519123445/j2li1tz9.jpg');
background-size: cover;
background-position: center;
background-repeat: no-repeat;
color: white;
font-family: Arial, sans-serif;
display: flex;
flex-direction: column;
align-items: center;
margin: 0;
height: 100vh;
position: relative;
}
```

```
.overlay {  
  position: absolute;  
  top: 0;  
  left: 0;  
  width: 100%;  
  height: 100%;  
  background-color: rgba(0, 0, 0, 0.5);  
  z-index: 1;  
}  
.content {  
  position: relative;  
  z-index: 2;  
  display: flex;  
  flex-direction: column;  
  align-items: flex-start;  
  margin-left: 20px;  
  width: 100%;  
}
```

```
.header {  
  display: flex;  
  align-items: center;  
  justify-content: flex-end;  
  width: 100%;  
  padding: 10px 30px;  
  box-sizing: border-box;  
}  
.button-container {  
  display: flex;  
  gap: 20px;  
}  
.button {  
  color: white;  
  font-size: 0.8em;  
  padding: 5px 10px;  
  border: none;  
  background: none;  
  cursor: pointer;  
  text-decoration: none;  
}
```

```
.home-left {  
  display: flex;  
  flex-direction: column;  
  align-items: flex-start;  
  margin-top: 60px;  
  margin-left: 140px;  
}  
.logo {  
  display: flex;  
  align-items: center;  
  font-family: 'Lora', sans-serif;  
  margin-right: 100px;  
  margin-top: 20px;  
}  
.vital {  
  font-weight: normal;  
  font-size: 6em;  
  margin-right: -56.8px;  
}
```

```
.track {  
font-weight: normal;  
font-size: 5.5em;  
margin-top: -10px;  
}  
.vital-track-container img {  
width: 60px;  
height: auto;  
margin-left: -20px;  
margin-top: -20px;  
z-index: 1;  
}  
.real-time-info {  
color: lightgray;  
font-size: 15px;  
margin-top: 90px;  
margin-left: -180px;  
}
```

```
.data-container {  
display: flex;  
flex-direction: row;  
gap: 20px;  
margin-top: 70px;  
margin-left: 30px;  
}  
.data-column {  
display: flex;  
flex-direction: column;  
gap: 60px;  
}  
.data-item {  
font-size: 1em;  
text-align: left;  
}  
#box4 {  
margin-left: 80px;  
}
```

```
#box5 {  
margin-left: 80px;  
}  
#box6 {  
margin-left: 80px;  
}  
.box {  
display: flex;  
flex-direction: column;  
align-items: flex-start;  
}  
.white-bar {  
width: 5px;  
height: 100%;  
background-color: white;  
margin-right: 20px;  
}
```

```

.map {
position: absolute;
top: 120px;
right: 0;
left: 850px;
z-index: 2;
font-size: 10px;
}
.search-container {
display: flex;
align-items: center;
gap: 10px;
margin-bottom: 10px;
}
.search-input {
padding: 5px;
border-radius: 3px;
border: 1px solid #ccc;
font-size: 1em;
}

```

```

.search-button {
padding: 5px 10px;
border-radius: 3px;
border: none;
background-color: #007BFF;
color: white;
cursor: pointer;
}

```

```

</style>
<script src="/socket.io/socket.io.js"></script>
<script>
document.addEventListener('DOMContentLoaded', (event)
=> {
const socket = io('http://localhost:3000');
socket.on('FromAPI', (data) => {
document.getElementById('heart_rate').innerText =
`${data.heart_rate}`;
document.getElementById('spo2').innerText =
`${data.spo2}`;
document.getElementById('temperature').innerText =
`${data.temperature}`;
document.getElementById('sound').innerText =
`${data.sound}`;
document.getElementById('gas').innerText = `${data.gas}`;
});
});
</script>
</head>
<body>
<div class="overlay"></div>
<div class="content">
<div class="header">
<div class="button-container">
<a href="home.html" class="button">Home</a>
<a href="http://192.168.45.82:8080/stream.html"
class="button">Monitor</a>

```

```

</div>
</div>
<div class="home-left">
<div class="logo">
<span class="vital">VITAL</span><span
class="track">Track</span>
<div class="vital-track-container">

</div>
<div class="real-time-info">Real-time information</div>
</div>
<div class="data-container">
<div class="white-bar"></div>
<div class="data-column">
<div id="box1">
<div class="data-item">
<div class="data-value" id="heart_rate">Data not
available</div>
Heart_rate
</div>
</div>

```

```

<div id="box2">
<div class="data-item">
<div class="data-value"
id="temperature">Data not available</div>
Temperature
</div>
</div>
<div id="box3">
<div class="data-item">
<div class="data-value" id="spo2">Data
not available</div>
Oxygen
</div>
</div>
</div>
<div class="data-column">
<div id="box4">
<div class="data-item">
<div class="data-value" id="sound">Data
not available</div>
Sound
</div>
</div>
<div id="box5">
<div class="data-item">
<div class="data-value" id="gas">Data not
available</div>
Gas
</div>
</div>
</div>
</div>

```

```

</div>
</div>
<div class="map">
<div class="search-container">
<h1>Find Hospital</h1>
<input type="text" id="keyword" class="search-input"
placeholder="Enter location">
<button onclick="searchPlaces()" class="search-
button">Search</button>
</div>
<div id="map" style="width: 500px; height: 400px;"></div>
</div>
<script type="text/javascript"
src="//dapi.kakao.com/v2/maps/sdk.js?appkey=7a694cc44416b51
ab573da07282884e2&libraries=services"></script>
<script type="text/javascript">
var mapContainer = document.getElementById('map');
var mapOption = {
center: new kakao.maps.LatLng(37.58512180114156,
126.92503818995559),
level: 3
};

```

```

var map = new kakao.maps.Map(mapContainer, mapOption);
var ps = new kakao.maps.services.Places();
function searchPlaces() {
var keyword = document.getElementById("keyword").value;
if (!keyword.trim()) {
alert('Please enter a keyword!');
return;
}
ps.keywordSearch(keyword, placesSearchCB);
}
function placesSearchCB(data, status, pagination) {
if (status === kakao.maps.services.Status.OK) {
var bounds = new kakao.maps.LatLngBounds();
for (var i = 0; i < data.length; i++) {
displayMarker(data[i]);
bounds.extend(new kakao.maps.LatLng(data[i].y, data[i].x));
}
}

```

```

map.setBounds(bounds);
} else if (status ===
kakao.maps.services.Status.ZERO_RESULT) {
alert('No results found.');
```

```

} else if (status === kakao.maps.services.Status.ERROR) {
alert('An error occurred while searching.');
```

```

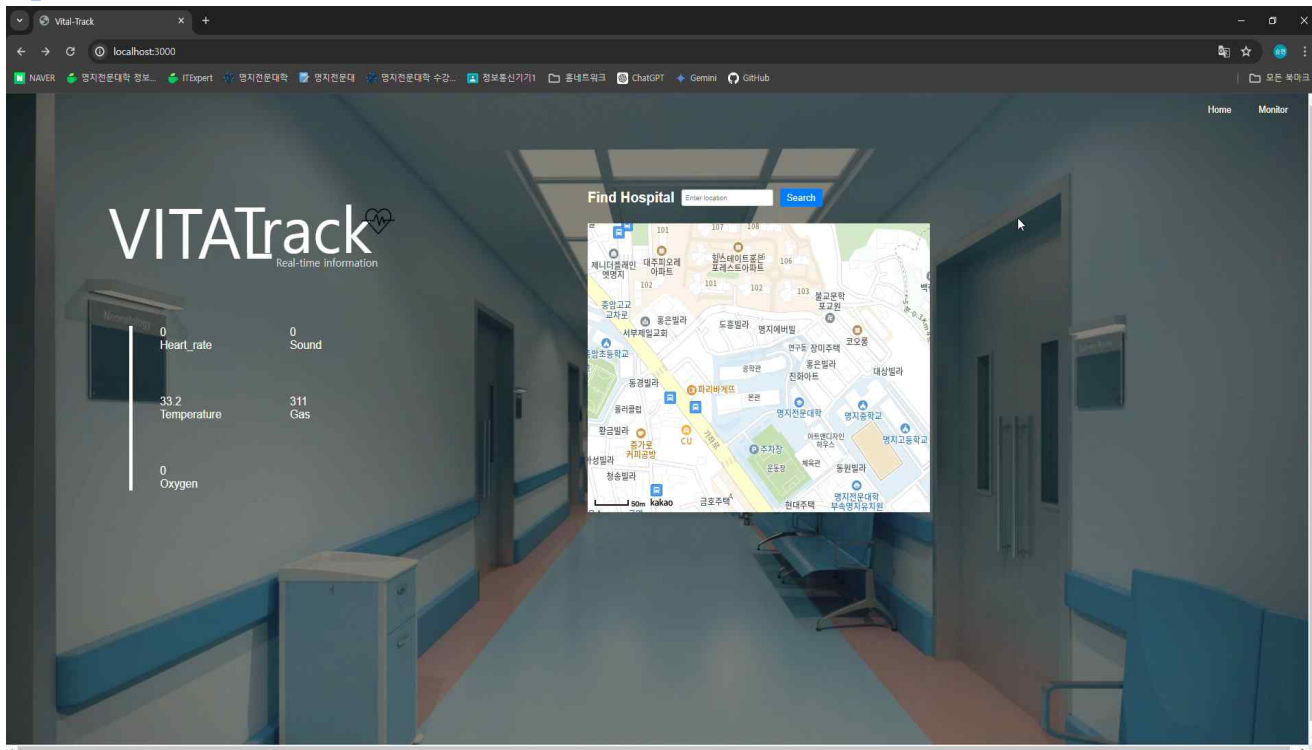
}
}
function displayMarker(place) {
var marker = new kakao.maps.Marker({
map: map,
position: new kakao.maps.LatLng(place.y, place.x)
});
kakao.maps.event.addListener(marker, 'click', function () {
var infowindow = new kakao.maps.InfoWindow({
content: '<div style="padding:5px;font-size:12px;">' +
place.place_name + '</div>'
});
infowindow.open(map, marker);
});
}
</script>
</body>
</html>

```


실행 결과

웹 화면

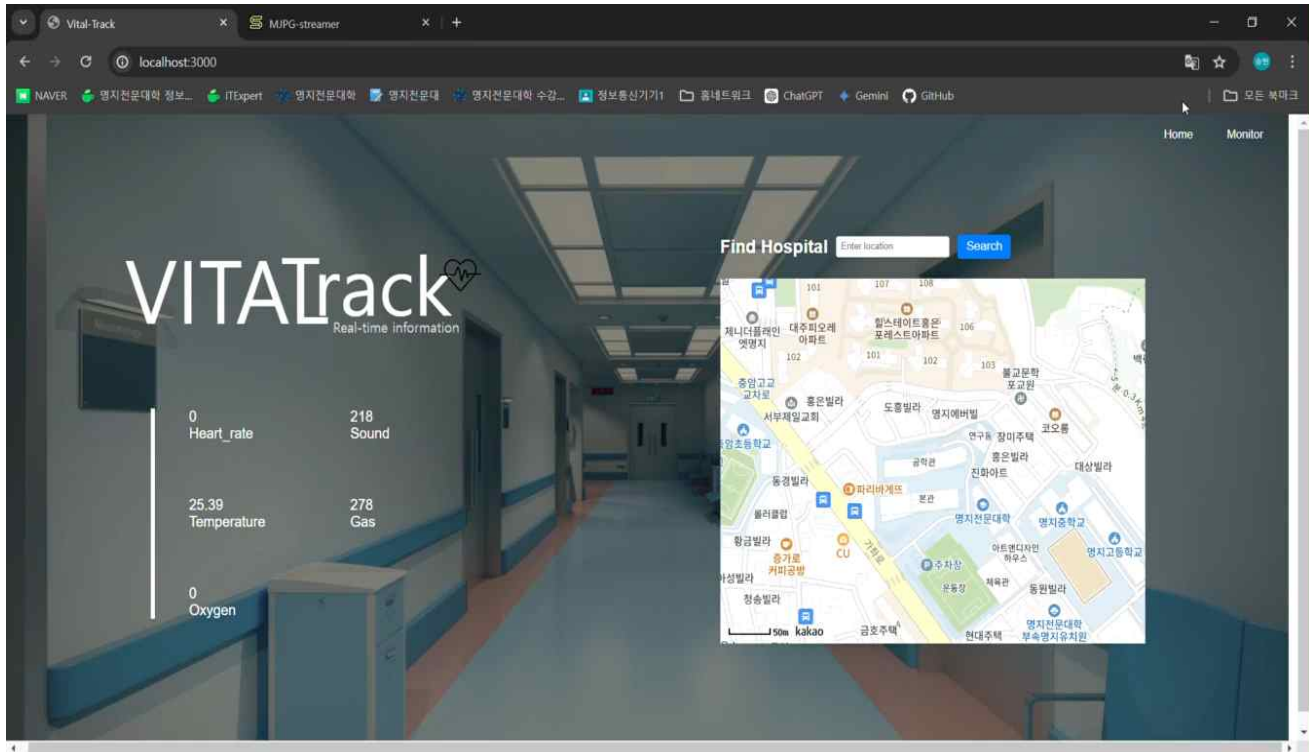
(<http://localhost:3000/>)



실행 결과

웹 화면

(http://localhost:3000/)



앱 화면 설계

Expo CLI 설치

```
npm install expo-cli --global
```

Expo 프로젝트 생성

```
Expo install expoProject
```

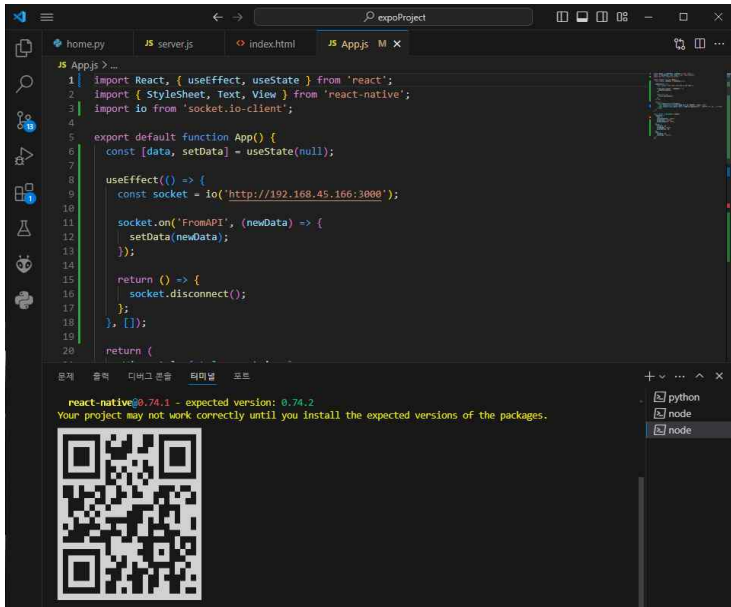
App.js

```
import React, { useEffect, useState } from 'react';
import { StyleSheet, Text, View } from 'react-native';
import io from 'socket.io-client';
export default function App() {
  const [data, setData] = useState(null);
  useEffect(() => {
    const socket = io('http://192.168.45.34:3000');
    socket.on('FromAPI', (newData) => {
      setData(newData);
    });
    return () => {
      socket.disconnect();
    };
  }, []);
  return (
    <View style={styles.container}>
      <Text style={styles.title}>환자 감시 IoT 모니터링 시스템</Text>
      <Text style={styles.data}>{data ? JSON.stringify(data) : '데이터 로드 중...'}</Text>
    </View>
  );
}
```

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#fff',
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
    marginBottom: 20,
  },
  data: {
    fontSize: 16,
    textAlign: 'center',
  },
});
```

실행 결과


앱 화면



```
1 | import React, { useEffect, useState } from 'react';
2 | import { StyleSheet, Text, View } from 'react-native';
3 | import io from 'socket.io-client';
4 |
5 | export default function App() {
6 |   const [data, setData] = useState(null);
7 |
8 |   useEffect(() => {
9 |     const socket = io('http://192.168.45.166:3000');
10 |
11 |     socket.on('FromAPI', (newData) => {
12 |       setData(newData);
13 |     });
14 |
15 |     return () => {
16 |       socket.disconnect();
17 |     };
18 |   }, []);
19 |
20 |   return (

```

react-native@0.74.1 - expected version: 0.74.2
Your project may not work correctly until you install the expected versions of the packages.



00:06
카메라



환자 감시 IoT 모니터링 시스템

```
{
  "id": "504",
  "heart_rate": 0,
  "spo2": 0,
  "temperature": 31.25,
  "sound": 106,
  "gas": 106,
  "measured_at": "2024-06-08T15:06:47.000Z"
}
```

웹캠 스트리밍

패키지 설치

MJPEG-Streamer 설치

Sudo apt update

Sudo apt install libjpeg-dev imagemagick libv4l-dev

// Git 설치

sudo apt install git

// MJPG-Streamer 소스 코드 다운로드

git clone <https://github.com/jacksonliam/mjpg-streamer.git>

// MJPG-Streamer 빌드

cd mjpg-streamer/mjpg-streamer-experimental

make

sudo make install

ufw 설치

Sudo apt install ufw

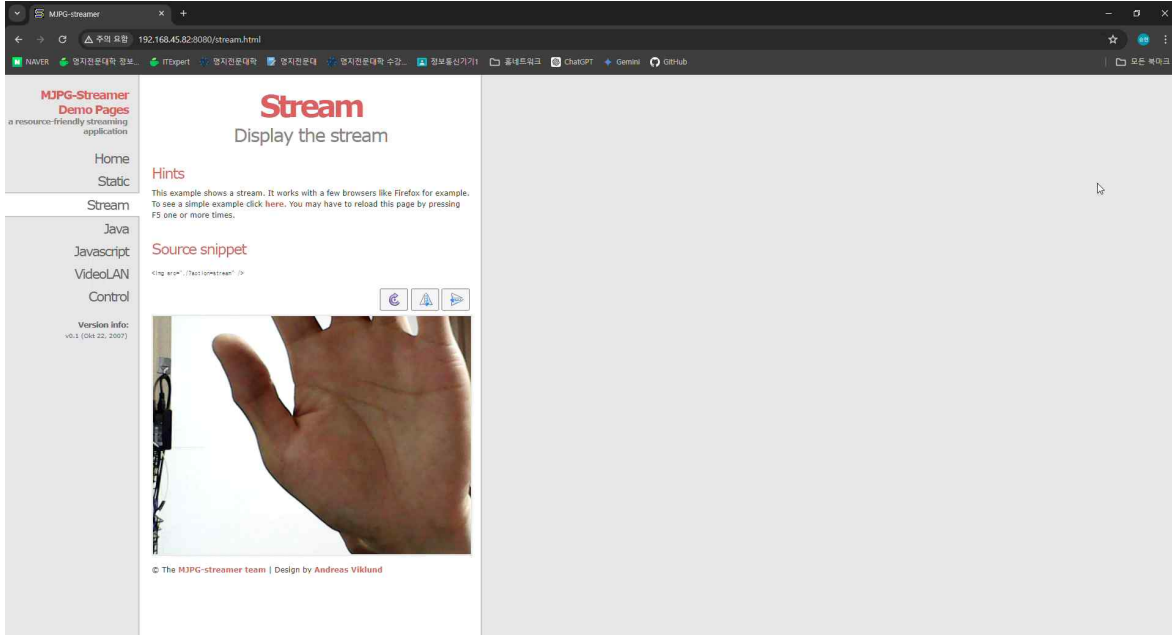
방화벽 비활성화

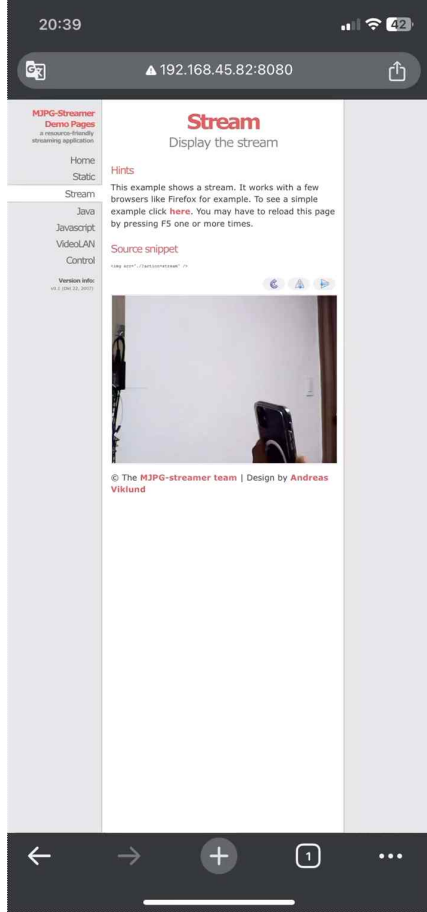
Sudo ufw disable

MJPEG- Streamer 실행

**./mjpg_streamer -i "/input_uvc.so -d /dev/video0 -r 640x480 -f 30"
-o "/output_http.so -p 8080 -w ./www"**





웹캠 실시간 스트리밍(<http://192.168.79.82:8080/stream.html>)





웹캠 실시간 스트리밍(<http://192.168.79.82:8080/stream.html>)
(모바일)

활용기자재

사진	제품명	제품번호	단가	부가세	구매가	배송	배송기간	URL
	아두이노 메 가2560 (R3) 호환보드	SZH-EK028	22,000원	2,200원	24,200원	국내	1~2일	https://www.devicemart.co.kr/goods/view?no=1278958
	아두이노 메 가2560 이지 모듈 확장 설 드	SZH-EK079	26,000원	2,600원	28,600원	국내	2~3일	https://www.devicemart.co.kr/goods/view?no=1314313
	MAX30102 산소포화도, 심박 센서 모 듈 (블랙)	SZH-JA057	1,700원	170원	1,870원	국내	1~2일	https://www.devicemart.co.kr/goods/view?no=13137639
	LM35 선형 아날로그 온 도 센서 모듈	SEN030002	9,600원	960원	10,560원	해외	1주일	https://www.devicemart.co.kr/goods/view?no=1378312

활용 기자재 “

	아두이노 고 감도 <u>샤운드</u> 센서 모듈	SEN050302	4,200원	420원	4,620원	해외	3~4일	https://www.devicemart.co.kr/goods/view?no=1289314
	아두이노 가 연성 가스 센 서 모듈	SZH-SSBH-0 26	1,400원	140원	1,540원	국내	2~3일	https://www.devicemart.co.kr/goods/view?no=1327418
	<u>라즈베리파이</u> 4	Raspberry Pi 4 Model B 8GB	100,000원	10,000원	110,000원	국내	1~2일	https://www.devicemart.co.kr/goods/view?no=12553062
	<u>화상카메라</u>	1035663	4,546원	454원	5,000원	국내	2일	https://www.daisomall.co.kr/pd/pdr/SCR_PDR_0001?pdNo=1035663&reclmYn=N

아쉬운 점

“

1. 비접촉 온도센서 모듈 미사용

2. 앱 화면 설계 불충분

3. 환자 건강 상태 이상 발생 시 알림 기능 미구현

감사합니다.