Be Attention

B조(집중해조)

강세희
김성수
서예진
이진아
최용욱

https://github.com/KHU-Rainbow

INDEX

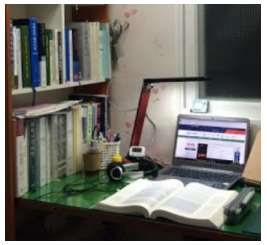
Part 0,

Overview

Part 0,

Overview

Overview



책상 앞에서 꿈을 키워 나가는 사람을 위한 프로젝트

Back end

공부 시간 자동 측정

핸드폰 사용 횟수 측정

AWS RDS, Lambda 구축

Rainbow

Front end

목표 공부 시간 입력

목표 성취 여부 통계 시각화

한 눈에 볼 수 있는 달력형 UI

Part 1,

Goal/Problem & Requirement

Part 1,

Goal/Problem & Requirement

Goal/Problem

사용자가 핸드폰으로
네트워크에 접속했는지 확인

사용자가 책상에
제대로 앉아있는지 감지

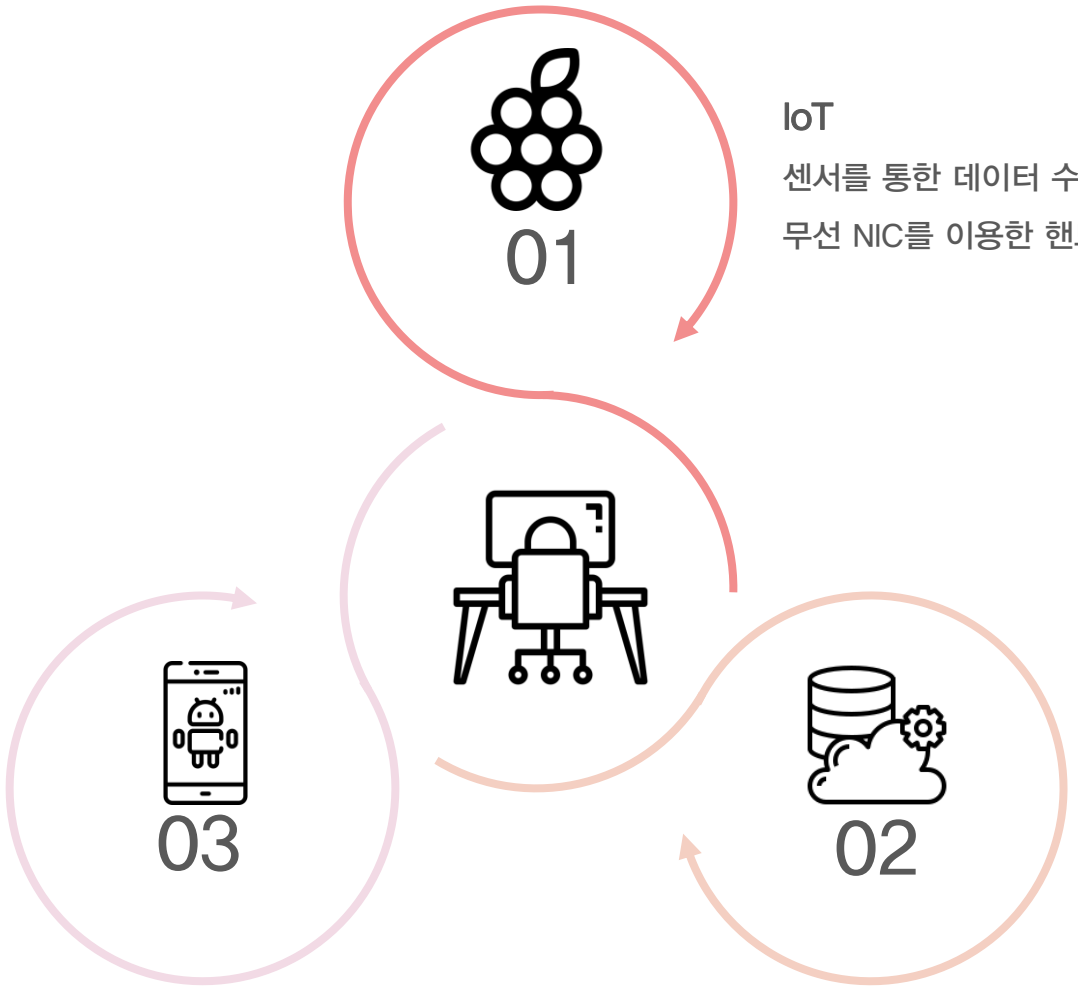
현재까지 누적된
공부 시간 출력

사용자의 총 공부 시간과
핸드폰 사용 횟수 알림

Part 1,

Goal/Problem & Requirement

Requirement



IoT

센서를 통한 데이터 수집 & 출력

무선 NIC를 이용한 핸드폰 패킷 감지

AWS Cloud

IoT 데이터를 AWS Cloud에 저장

Lambda 함수를 이용한 데이터 처리

Application

AWS Cloud 데이터 시각화

한 눈에 볼 수 있는 달력형 UI

Part 2,

Approach

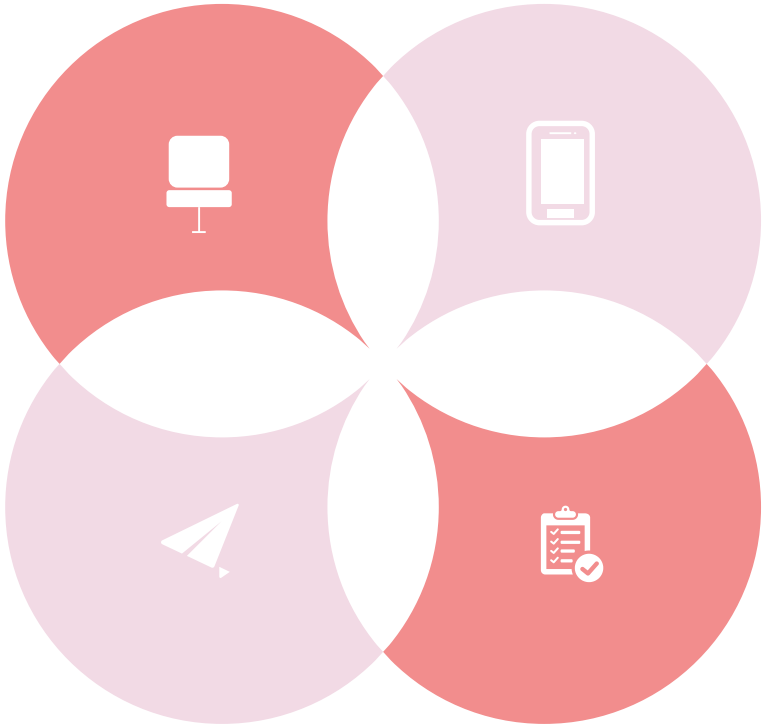
Part 2,

Approach

Approach: A, B, C, D

Approach A
사용자가 책상에 앉아있는
실제 공부 시간 측정

Approach B
공부 중간에 핸드폰을
사용하려 시도한 횟수 측정



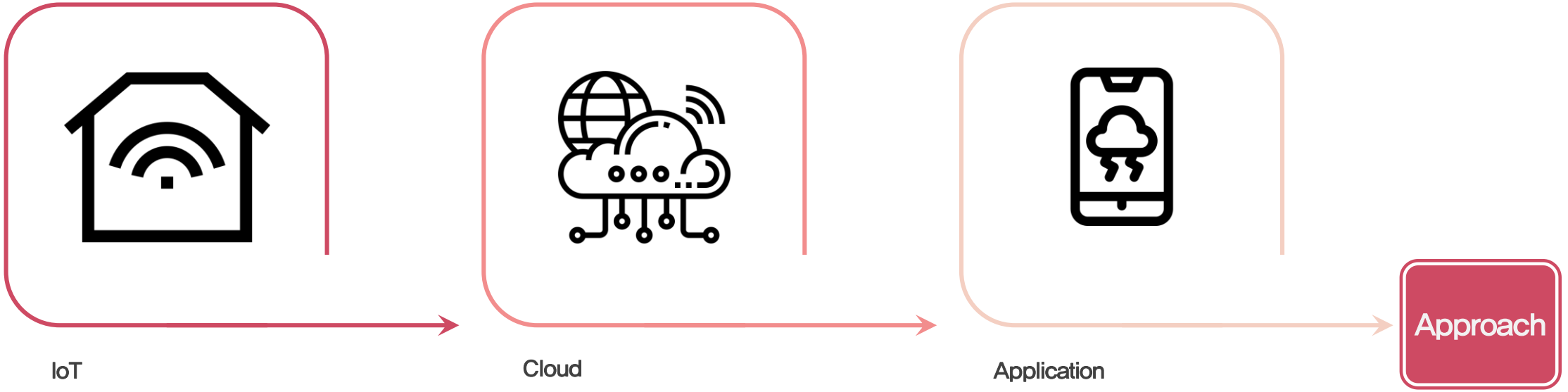
Approach C
센서에서 측정한 데이터를
서버에 저장하는 과정 자동화

Approach D
서버의 데이터를 바탕으로
App에서 공부 관련 통계 시각화

Part 2,

Approach

Approach: Development Scheme



IoT

센서를 통한 데이터 수집

데이터 저장 과정 자동화

Cloud

AWS Cloud DB 구축

Lambda 함수로 데이터 처리

Application

AWS Cloud 데이터 시각화

공부 시간, 핸드폰 사용 횟수 등 통계

Part 3,

Development Environment

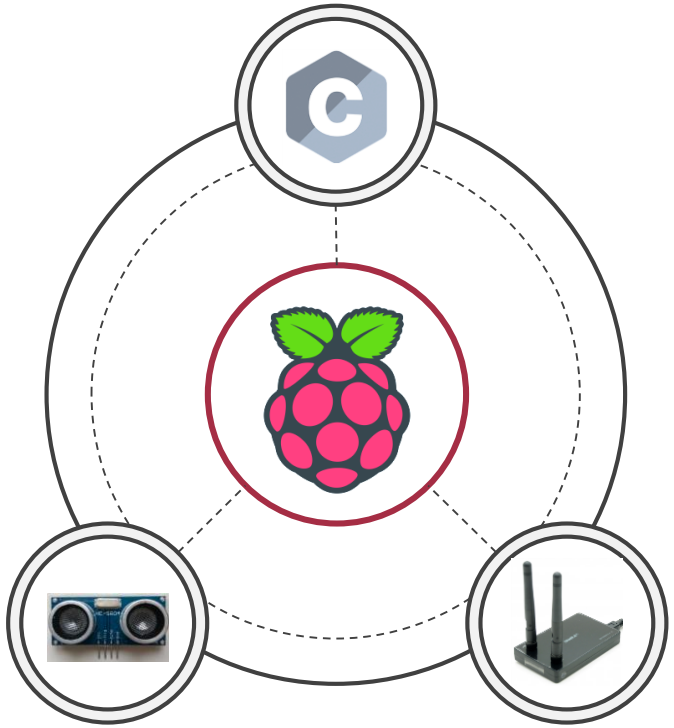
Part 3,

Development Environment

Development Environment: RPi

프로그램

Python, C



센서

초음파, Buzzer, FND

핸드폰 패킷 감지

무선 네트워크 어댑터

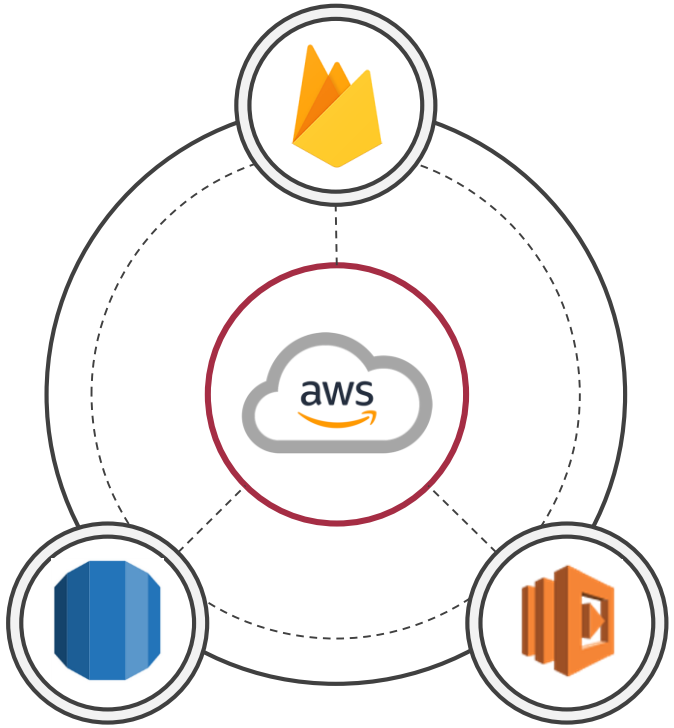
Part 3,

Development Environment

Development Environment: Cloud

Push 알림

Google Firebase



데이터베이스

AWS RDS

서버리스 컴퓨팅

AWS Lambda

Part 3,

Development Environment

Development Environment: Application

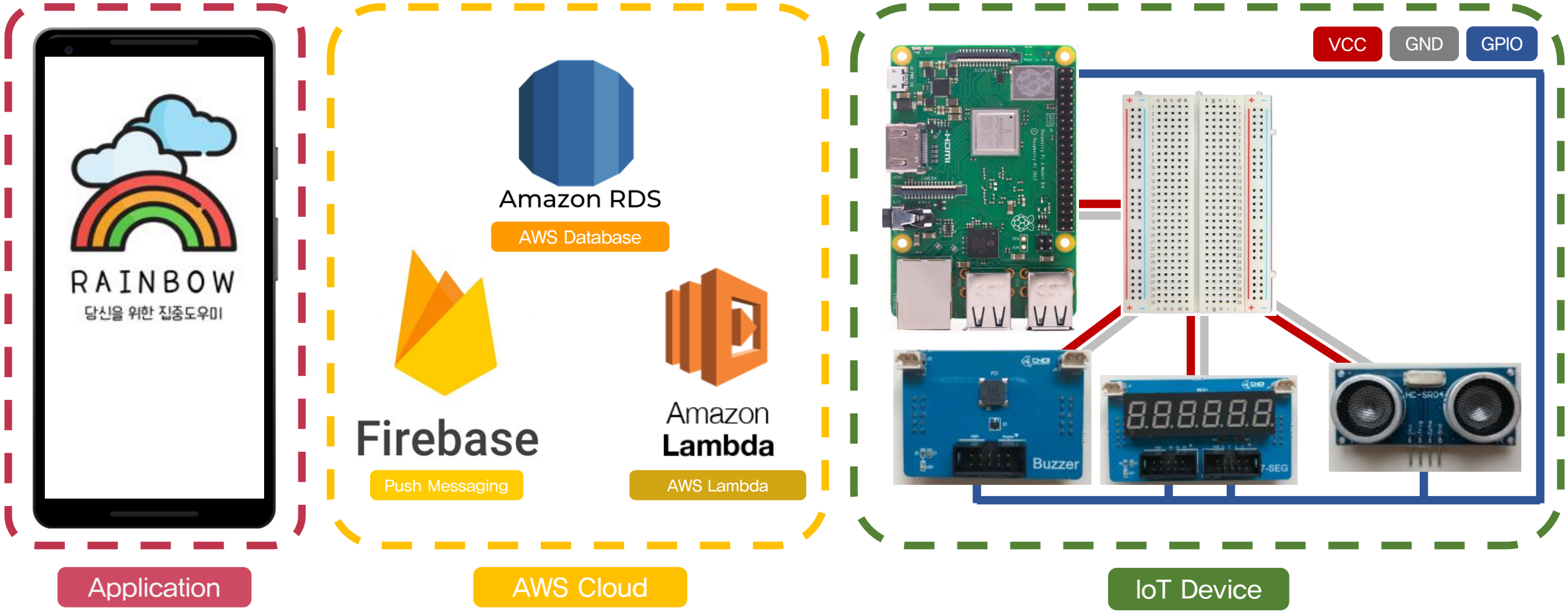
Part 4,

Architecture

Part 4,

Architecture

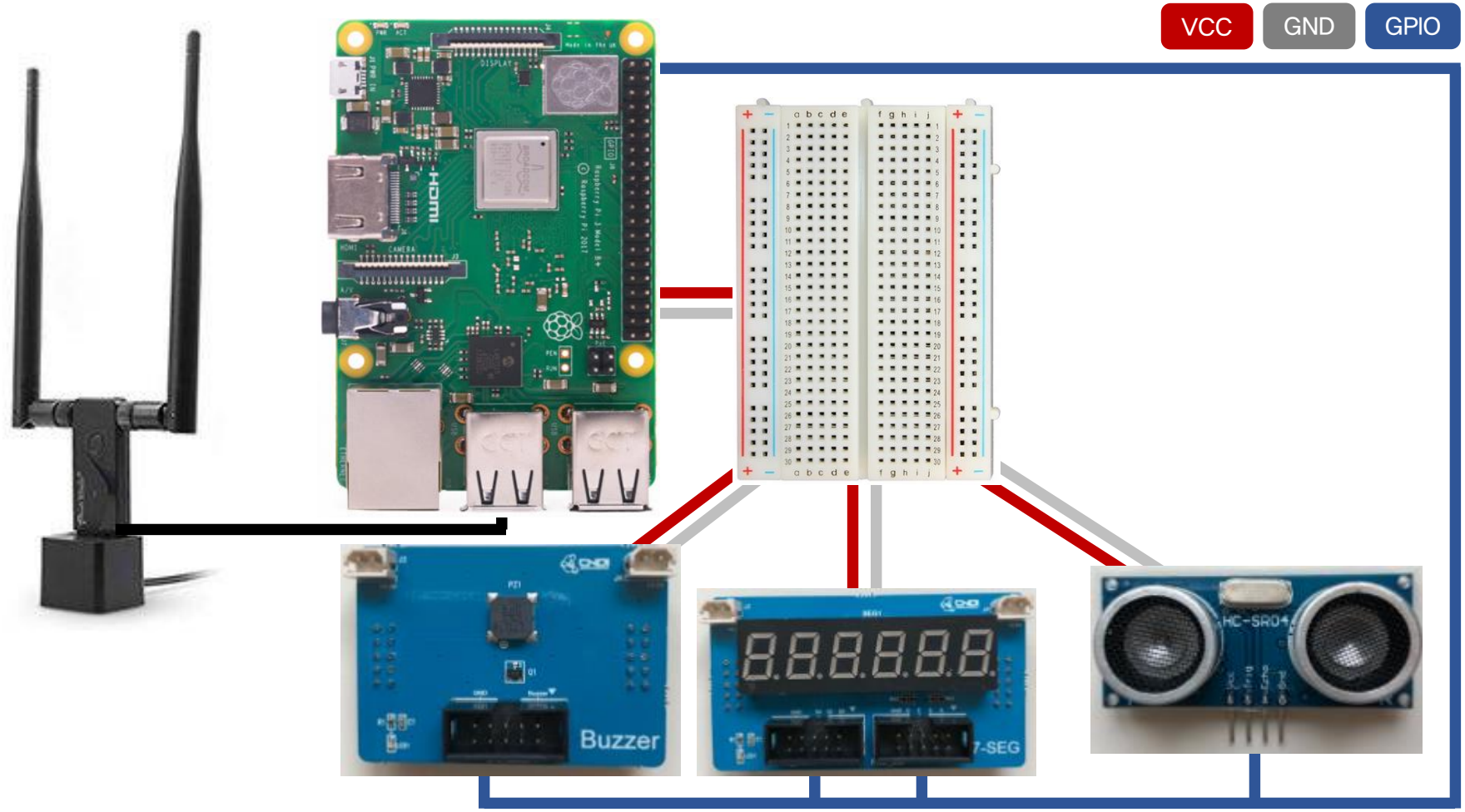
Overall Architecture



Part 4,

Architecture

IoT Device



Part 4,

Architecture

AWS Cloud

Amazon RDS

AWS Database

Firebase

Push Messaging

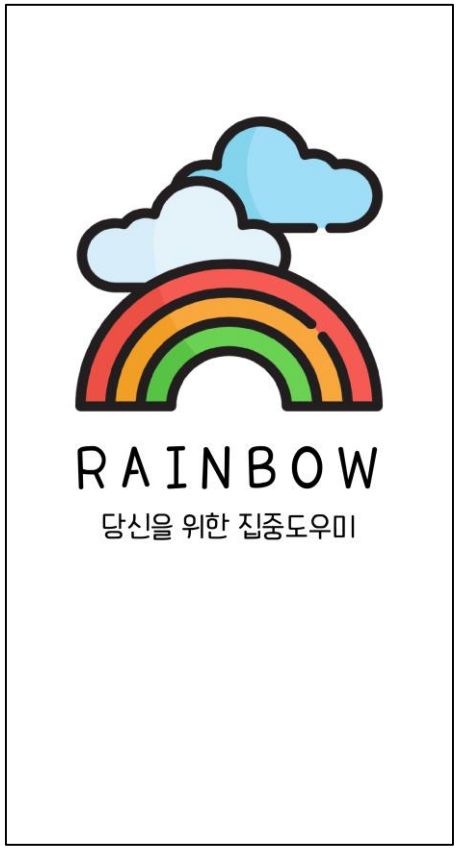
Amazon
Lambda

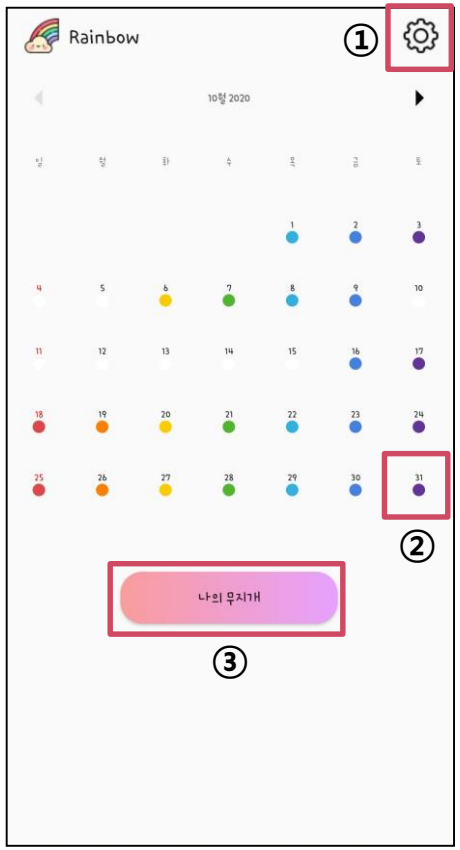
AWS Lambda

Part 4,

Architecture

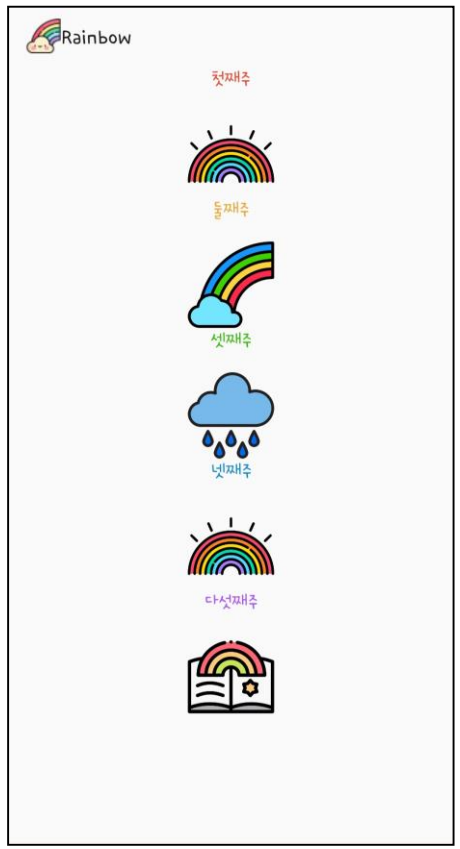
Application





①

②



③

Part 5,

Implementation Spec

Part 5,

Implementation Spec

Application
- Android

AWS Cloud
- RDS
- Lambda
- Firebase

IoT Device
- RPi
- Sensors

① 당일 목표 시간 설정
Value: time_target (int)

② Rpi Polling → 핸드폰 패킷 감지
Value: is_wifi (bool)

If) is_wifi == true

부저 센서 → 삐빅♬ 소리 + wifi 카운트
Value: wifi_count (int)

③ 초음파 센서 → 앉은 자세 감지
Value: is_sit (bool)

If) is_sit == true

부저 센서 → 도미솔♬ 소리

FND 센서 → 시간 카운팅 시작
Value: time_count (int)

Part 5,

Implementation Spec

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

# Run ~/ASM/SFU_OTA/crontab.py Everyday 00:00 AM
#0 0 * * * /home/pi/venv/bin/python3 /home/pi/ASM/SFU_OTA/crontab.py
0 0 * * * /home/pi/venv/bin/python3 /home/pi/School/Project/Study/study.py
```

④ 초음파 센서 → 일어남 감지

If) is_sit == false

부저 센서 → 솔미도♬ 소리

FND 센서 → 시간 카운팅 스탑

⑤ 매일 자정 → Crontab 이용 결과 전달
Return: time_count, wifi_count

⑥ Lambda → Data Update
Value: time_count, wifi_count

⑦ Rpi → Data Initialize
Value: time_count, wifi_count

Part 5,

Implementation Spec

⑧ Lambda, Firebase → Push 알림 전송
Return: time_target, time_count, wifi_count

⑨ Lamda: 목표/실제 공부 시간 비교

If) time_target <= time_count

UI → 동그라미 지워짐
Resource: red, yellow, · · · purple

⑩ Lambda: 동그라미 버튼 클릭 이벤트

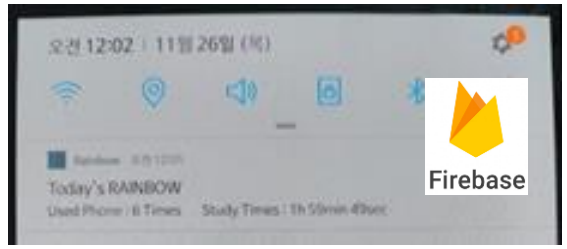
OnClick) SQL Query로 record 호출

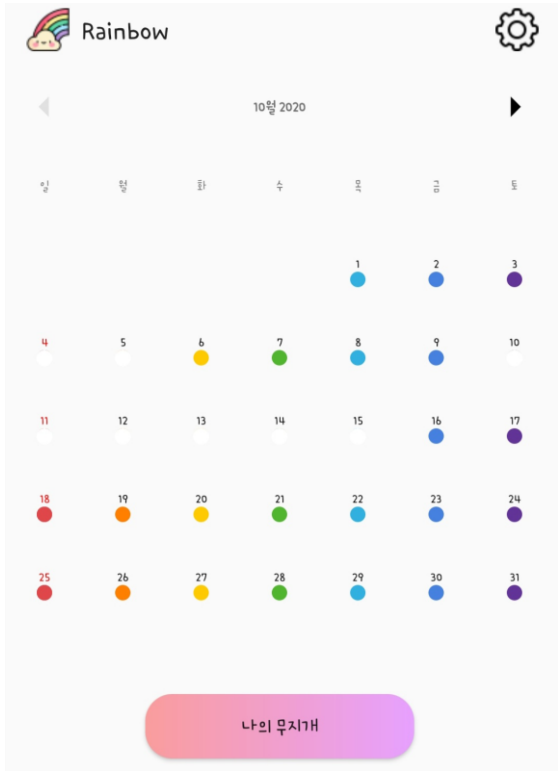
공부 시간 Time-line UI로 출력
Value: time_target time_count, wifi_count

⑪ Lambda: 1주일간 동그라미 수 측정

SQL Query) today == sunday

UI → Rainbow 채워짐
Resource: rb0, rb1, · · · , rb7





Part 6,

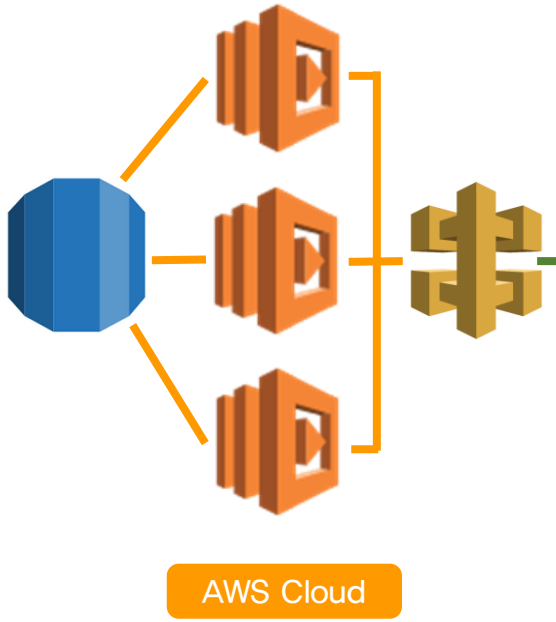
Result

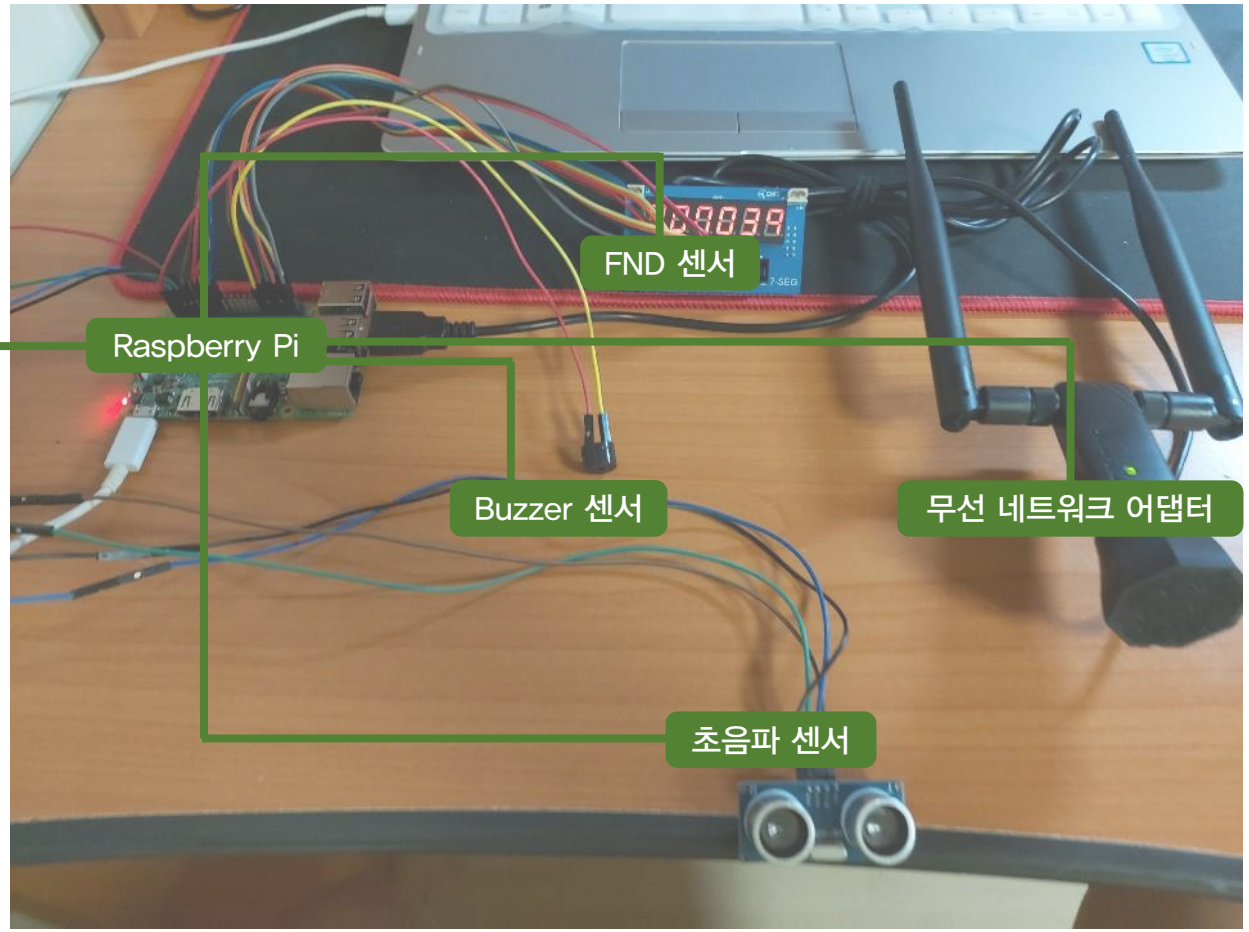
Part 6,

Result

IoT | Cloud | App

Result: IoT





Part 6,

Result

IoT | Cloud | App

Result: IoT

```
//Adapter model name : AWUS036NH(ALFA), PAU09(PANDA)
//length : 18
struct radiotap_header {
        uint8_t         it_version;     /* set to
        uint8_t         it_pad;
        uint16_t        it_length;      /* entire
        uint32_t        it_present_flags;       /* fields
        uint8_t         it_flags;
        uint8_t         it_data_Rate;
        uint16_t        it_channel_frequency;
        uint16_t        it_channel_flags;
        uint8_t         it_antenna_signal;
        uint8_t         it_antenna;
        uint16_t        it_RX_flags;
};
```

판다 네트워크 전용 라디오 탭 헤더

```
struct ieee80211_header {
        uint8_t         type_subtype;
        uint8_t         flags;
        uint16_t        duration;
        /*
         Beacon Frame, Probe Request, Probe Response, Authentication, Deauthentication,
            add1 = Receiver, Destination
            add2 = Transmitter, Source
            add3 = BSSID
         Data
            add1 = Rec, Des, STA
            add2 = Trans, BSSID
            add3 = Source
         Qos Null function, Qos Data, Null function
            add1 = Receiver, BSSID
            add2 = Transmitter, Source, STA
            add3 = Destination
          */
        uint8_t add1[6];
        uint8_t add2[6];
        uint8_t add3[6];
        uint16_t        fragment_sequence;
};
```

Probe Request 패킷 구분

MAC 정보 알아내는 부분

Part 6,

Result

IoT | Cloud | App

Result: IoT

```
void parsing(const u_char* packet, char* node_mac, char* my_device_mac){
    //packet header setting
    struct radiotap_header *rh = (struct radiotap_header *)packet;
    struct ieee80211_header *ih = (struct ieee80211_header *)(packet + rh->it_length);
    //uint8_t *wlh = (uint8_t *)ih + IEEE_LEN;              //wireless LAN header

    //my_device_mac change to hex
    uint8_t my_mac[6];
    char t[4];

    for(int i = 0; i <6; i++){
        memcpy(t, (my_device_mac +i*3), 3);
        t[3] = '\0';
        *(my_mac +i) = (uint8_t)strtoul(t, NULL, 16);
    }
    //printf("my_mac :  %02x:%02x:%02x:%02x:%02x:%02x   ",my_mac[0], my_mac[1], my_mac[2], my_mac[3], my_mac[4], my_mac[5]);

    //Catch Probe_request & parsing
    if( ih->type_subtype == PROBE_REQUEST && rh->it_antenna_signal > 186){
        static int cnt = 0;
        cnt++;
        printf("%3d Probe Request=============================================\n", cnt);
        ledControl();

        char device_mac[17]; // enough size
        //%02X : 2 hex code
        sprintf(device_mac, "%02x:%02x:%02x:%02x:%02x:%02x", ih->add2[0], ih->add2[1], ih->add2[2], ih->add2[3], ih->add2[4], ih->add2[5]);
        printf("device_mac : %s, ", device_mac);

        if((my_mac[0] == ih->add2[0])&&(my_mac[1] == ih->add2[1])&&(my_mac[2] == ih->add2[2])
            &&(my_mac[3] == ih->add2[3])&&(my_mac[4] == ih->add2[4])&&(my_mac[5] == ih->add2[5])){
            printf("\n=== This is from your phone ===\n");
```

라디오 탭, ieee 802.11 Header 부분 가져오기

MAC 주소 비교를 위한 형 변환(uint8_t)

lt_antenna_signal 〉 186
(신호세기 일정 수준 이상인 것만 캡쳐)

MAC 주소 비교를 위한 형 변환 (char)

MAC 주소 비교

Part 6,

Result

IoT | Cloud | App

Result: IoT

```
//Send Data to Cloud Service
CURL *curl;
CURLcode res;

std::string strTargetURL;
std::string strResourceJSON;
std::string s_today(today);

struct curl_slist *headerlist = nullptr;
headerlist = curl_slist_append(headerlist, "Content-Type: application/json");

strTargetURL = "https://r89kbtj8x9.execute-api.us-east-1.amazonaws.com/dev/detect";
strResourceJSON = "{\"Packet_date\": \"" + s_today + "\", " + "\"Packet_time\": \"" + std::to_string(p_data) +"\"}";

curl_global_init(CURL_GLOBAL_ALL);
curl = curl_easy_init();

if (curl)
{
    curl_easy_setopt(curl, CURLOPT_URL, strTargetURL.c_str());
    curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headerlist);
    curl_easy_setopt(curl, CURLOPT_SSL_VERIFYPEER, false);
    curl_easy_setopt(curl, CURLOPT_SSL_VERIFYHOST, false);
    curl_easy_setopt(curl, CURLOPT_POST, 1L);
    curl_easy_setopt(curl, CURLOPT_POSTFIELDS, strResourceJSON.c_str());

    res = curl_easy_perform(curl);

    curl_easy_cleanup(curl);
    curl_slist_free_all(headerlist);
}
```

오늘 날짜, 핸드폰 사용 감지 시각 json 형식으로 만들기

curl 라이브러리 이용, AWS RDS에 업로드

Result

IoT | Cloud | App

Result: IoT

```
pi@raspberrypi:~/School/Project/Packet $ ls
dependencies.sh  getmac.o  main.o    packet      parser.cpp  pkt.h
getmac.cpp       main.cpp  Makefile  packet.pro  parser.o    radiotap_header.h
```

```
device_mac : 14:bd:61:ef:d1:0c, rssi : -50, timestamp : 1605679543

 42 Probe Request============================================
device_mac : 50:50:a4:0e:16:90,
=== This is from your phone ===
BuzzerFlag is 1
DETECT_DATE: 2020-11-18
DETECT_TIME: 21969

rssi : -48, timestamp : 1605679570

 43 Probe Request============================================
device_mac : 50:50:a4:0e:16:90,
=== This is from your phone ===
BuzzerFlag is 1
DETECT_DATE: 2020-11-18
DETECT_TIME: 21971

rssi : -48, timestamp : 1605679572
```

핸드폰의 MAC 주소

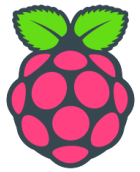
패킷 발생 시각 및 Buzzer Sensor 작동

패킷의 신호 세기

packet.exe 실행 화면

Part 6,

Result



Result: IoT

IoT | Cloud | App

```
// 초음파 센서를 이용해 Distance를 받아오는 함수
double getDistance() {
    double fDistance = 0.0;
    int nStartTime, nEndTime;
    nStartTime = nEndTime = 0;

    digitalWrite(TP, LOW);

    delayMicroseconds(10);
    digitalWrite(TP, HIGH);

    delayMicroseconds(10);
    digitalWrite(TP, LOW);

    while (digitalRead(EP) == LOW);
    nStartTime = micros();

    while (digitalRead(EP) == HIGH);
    nEndTime = micros();

    fDistance = (nEndTime - nStartTime) / 29. / 2.

    return fDistance;
}
```

RPi, 초음파 센서 연동

```
// FND를 선택하는 함수, S0 ~ S5 중 파라미터(position)에 해당하는 FND 선택
void FndSelect(int position) {
    int i;
    for (i = 0; i < 6; i++) {
        if (i == position) {
            digitalWrite(FndSelectPin[i], LOW); // 선택된 FND의 Select 핀 ON
        }
        else {
            digitalWrite(FndSelectPin[i], HIGH); // 선택되지 않은 FND의 Select 핀 OFF
        }
    }
}

// FND를 출력하는 함수
void FndDisplay(int position, int num) {
    int i, j;
    int flag = 0;     // FndPin[ ]을 ON/OFF
    int shift = 0x01; // FndFont와 And 연산하여 출력할 LED의 상태 결정
    for (i = 0; i < 8; i++) {
        flag = (FndFont[num] & shift); // i = 0, FndFont[ 0 ] = 0x3F라 하면 (0b00111111 & 0b00000100 = 1) 이다.
        digitalWrite(FndPin[i], flag); // FndPin[ ]을 flag( 0또는 1 )로 ON/OFF
        shift <<= 1;                   // 왼쪽으로 한 비트 쉬프트한다. I = 0이라 하면, ( shift = 0b00000001 )에서
    }

    FndSelect(position);
}
```

RPi, FND 센서 연동

Part 6,

Result

IoT | Cloud | App

Result: IoT

```
// 초음파 센서 감지, 버튼 딩동댕
if(is_sit == 0){
    if (f1 < 30.0 || f2 < 30.0 || f3 < 30.0 || f4 < 30.0 || f5 < 30.0) {
        is_sit = 1;
        softToneWrite(BP, melody[0]);
        delay(250);
        softToneWrite(BP, melody[2]);
        delay(250);
        softToneWrite(BP, melody[4]);
        delay(250);
        softToneWrite(BP, 0);
        printf("\nDistance: %8.4f, %8.4f, %8.4f, %8.4f, %8.4f", f1, f2, f3, f4, f5);
    }
}

// 초음파 센서 감지, 버튼 댕동딩
else if (is_sit == 1){
    if (f1 > 30.0 && f2 > 30.0 && f3 > 30.0 & f4 > 30.0 && f5 > 30.0) {
        is_sit = 0;
        softToneWrite(BP, melody[4]);
        delay(250);
        softToneWrite(BP, melody[2]);
        delay(250);
        softToneWrite(BP, melody[0]);
        delay(250);
        softToneWrite(BP, 0);
```

RPi, Buzzer 센서 연동

```
// 객체에 키를 추가하고 공부날짜 저장
json_object_set_string(rootObject, "Study_date", today);
// 객체에 키를 추가하고 공부시간 저장
json_object_set_number(rootObject, "Study_time", s_data);

//Send Data to Cloud Service
printf("\nDistance: %8.4f, %8.4f, %8.4f, %8.4f, %8.4f", f1, f2, f3, f4, f5);
printf("\nSTUDY_DATE: %s", today);
printf("\nSTUDY_TIME: %d\n\n", s_data);

// JSON_Value를 사람이 읽기 쉬운 문자열(pretty)로 만든 뒤 파일에 저장
json_serialize_to_file_pretty(rootValue, "today.json");

// 객체에 키를 추가하고 공부날짜 저장
json_object_set_string(rootObject, "Study_date", tomorrow);
// 객체에 키를 추가하고 공부시간 저장
json_object_set_number(rootObject, "Study_time", 0);
json_object_set_number(rootObject, "Study_goal", 0);
json_object_set_number(rootObject, "Study_achieved", 0);

// JSON_Value를 사람이 읽기 쉬운 문자열(pretty)로 만든 뒤 파일에 저장
json_serialize_to_file_pretty(rootValue, "tomorrow.json");
```

공부 시간 json 파일 형태로 저장 & 누적

Part 6,

Result

IoT | Cloud | App

Result: IoT

```
pi@raspberrypi:~/School/Project/Study $ ls
Makefile  parson.h  study    study.o   today.json
parson.c  parson.o  study.c  study.py  tomorrow.json
```

```
pi@raspberrypi:~/School/Project/Study $ ./study

Distance:   5.7759,   5.8621,   5.8621,   5.8621,   5.8621

Distance: 164.4138, 163.5862, 164.8621, 164.4310, 164.4483

STUDY_DATE: 2020-11-30
STUDY_TIME: 9
```

today.json (~/School/Project/Study) - VIM

```
1 {
2     "Study_date": "2020-11-30",
3     "Study_time": 9
4 }
```

json 파일 업데이트

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

# Run ~/ASM/SFU_OTA/crontab.py Everyday 00:00 AM
#0 0 * * * /home/pi/venv/bin/python3 /home/pi/ASM/SFU_OTA/crontab.py
0 0 * * * /home/pi/venv/bin/python3 /home/pi/School/Project/Study/study.py
```

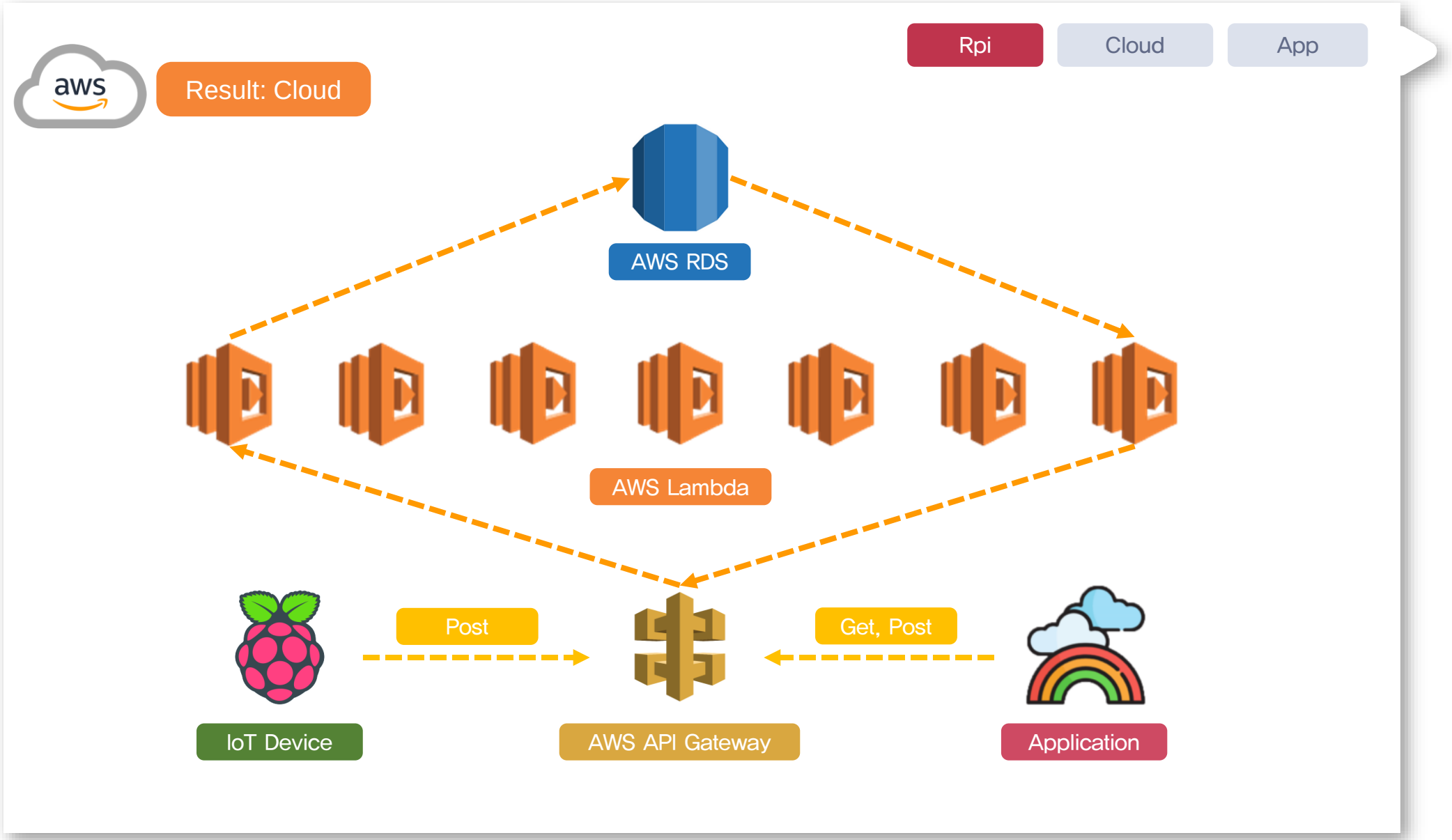
study.py (~/School/Project/Study) - VIM

```
import requests
import json

with open('/home/pi/School/Project/Study/today.json') as json_file:
    json_data = json.load(json_file)
    response = requests.post('https://r89kbtj8x9.execute-api.us-east-1.amazonaws.com

with open('/home/pi/School/Project/Study/tomorrow.json') as json_file:
    json_data = json.load(json_file)
    response = requests.post('https://r89kbtj8x9.execute-api.us-east-1.amazonaws.com
```

crontab 이용, AWS RDS 업로드

Part 6,

Result



Part 6,

Result

Rpi | Cloud | App

Result: Cloud

RDS > 데이터베이스 > rainbow-db-instance

rainbow-db-instance

요약

DB 식별자	CPU	상태
rainbow-db-instance	2.37%	사용 가능
역할	현재 활동	엔진
인스턴스	5 연결	MySQL Community

연결 & 보안 | 모니터링 | 로그 및 이벤트 | 구성 | 유지 관리 및 백업 | 태그

연결 & 보안

엔드포인트 및 포트

엔드포인트
rainbow-db-instance.cuggy6bbkwmy.us-east-1.rds.amazonaws.com

포트
3306

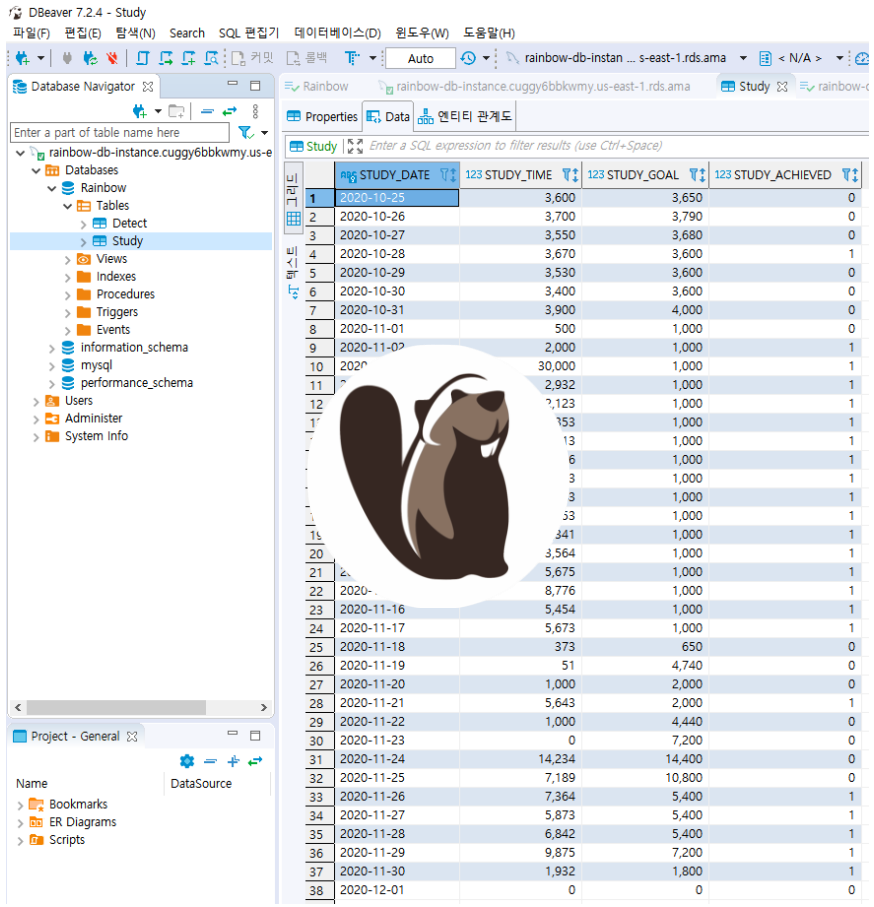
네트워킹

가용 영역
us-east-1a

VPC
rainbow-vpc (vpc-014cf03fad85eb8cf)

서브넷 그룹
default-vpc-014cf03fad85eb8cf

AWS RDS 구축



Study 테이블 데이터 (DBeaver)

Part 6,

Result

Rpi | Cloud | App

Result: Cloud

API > rainbow-API (r89kbtj8x9) > 리소스 > / (jpbue5m8xi)

리소스 작업 / 메서드

- /
  - /achievement
    - GET
  - /calendar
    - GET
  - /detect
    - POST
  - /estimation
    - GET
  - /goal
    - GET
    - POST
  - /interrupt
    - GET
  - /study-init
    - POST
  - /study-update
    - POST
  - /time
    - GET

AWS API Gateway 리소스

API > rainbow-API (r89kbtj8x9) > 스테이지 > dev > /achievement > GET

스테이지 생성

- dev
  - /
    - /achievement
      - GET
    - /calendar
      - GET
    - /detect
      - POST
    - /estimation
      - GET
    - /goal
      - POST
      - GET
    - /interrupt
      - GET
    - /rainbow-post-study
    - /study-init
      - POST
    - /study-update
      - POST
    - /time
      - GET

dev - GET - /achievement

URL 호출: https://r89kbtj8x9.execute-api.

이 페이지를 사용하여 GET의 /achievement 메서드에 대한 dev 스테이지 설정을 재정의합니다.

설정 ◉ 스테이지에서 상속

○ 이 메서드에 대해 재정의

URL 호출을 통해 Lambda 함수와 연결

Part 6,

Result

Rpi | Cloud | App

Result: Cloud

Lambda > 함수

함수 (10)

태그 및 속성별 필터 또는 키워드별 검색

함수 이름	설명	패키지 유형	런타임
rainbow-app-get-achieved	① 당일 목표 성취여부 계산 & 출력 기능	Zip	Python 3.7
rainbow-get-count_interrupt	② 핸드폰 사용 횟수 읽어오는 기능	Zip	Python 3.7
rainbow-get-estimation	③ 1주일 목표 성취여부 계산 & 출력 기능	Zip	Python 3.7
rainbow-get-goal	④ 목표 공부시간 읽어오는 기능	Zip	Python 3.7
rainbow-post-study-update	⑤ 당일 공부량 업데이트하는 기능	Zip	Python 3.7
rainbow-app-post-goal	⑥ 목표 공부시간 쓰는 기능	Zip	Python 3.7
rainbow-app-get-calendar	⑦ 월별 날짜 읽어오는 기능	Zip	Python 3.7
rainbow-post-detect	⑧ 핸드폰 사용 횟수 쓰는 기능	Zip	Python 3.7
rainbow-get-studytime	⑨ 당일 공부시간 읽어오는 기능	Zip	Python 3.7
rainbow-post-study-init	⑩ 다음 날짜 레코드 생성하는 기능	Zip	Python 3.7

AWS Lambda 함수 정의

Part 6,

Result

Rpi | Cloud | App

Result: Cloud

```
def lambda_handler(event, context):
    target_date = event["date"]
    goal_time = event["time"]
    # goal_time is
    with conn.cursor() as cur:
        sql = """UPDATE Study SET STUDY_GOAL = %s where STUDY_DATE = %s"""
        cur.execute(sql, (goal_time, target_date))
        conn.commit()
```

```
def lambda_handler(event, context):
    target_date = event['queryStringParameters']['target']
    cursor = conn.cursor()
    sql = "select STUDY_GOAL From Study where STUDY_DATE=%s"
    cursor.execute(sql,target_date)
    result = cursor.fetchall()
    print(result)
    if result[0][0] is None:
        date = str(target_date)
        year = date[0:4]
        month = date[5:7]
        day = date[8:]
        day = int(day)
        day -= 1
        if day == 0:
            month = int(month)
            if month == 1 or month == 3 or month == 5 or month == 7 or month == 8 or month == 10 or month == 12:
```

PyMySQL + Lamda → SQL Query 및 알고리즘 처리

IoT Device — Post → AWS API Gateway — Get, Post → Application

```
def find_sunday(year, month, day):
    if month == 1 or month == 3 or month == 5 or month == 7 or month == 8 or month == 10 or month == 12:
        if day >= 31:
            if month == 12:
                month = 1
                day = 1
            else:
                day = 1
    else:
        if day >= 28:
            month += 1
    if calendar.weekday(year,month,day) == 6:
        return (year, month, day)
    else:
        for i in range(6 - calendar.weekday(year,month,day)):
            if month == 1 or month == 3 or month == 5 or month == 7 or month == 8 or month == 10 or month == 12:
                if day == 31:
                    month += 1
                    day = 1
                else:
                    day += 1
            elif month == 4 or month == 6 or month == 9 or month == 11:
                if day == 30:
                    month += 1
                    day = 1
                else:
                    day += 1
            else:
                if day == 28:
                    month += 1
                else:
                    day += 1
        return (year, month, day)
```

```
start1 = find_sunday(year,month,day + 1)
end1 = find_saturday(start1[0], start1[1], start1[2])

startdate1 = convertdate(str(start1[0])) + "-" + convertdate(str(start1[1])) + "-" + convertdate(str(start1[2]))
enddate1 = convertdate(str(end1[0])) + "-" + convertdate(str(end1[1])) + "-" + convertdate(str(end1[2]))


start2 = find_sunday(end1[0], end1[1], end1[2])
end2 = find_saturday(start2[0], start2[1], start2[2])

startdate2 = convertdate(str(start2[0])) + "-" + convertdate(str(start2[1])) + "-" + convertdate(str(start2[2]))
enddate2 = convertdate(str(end2[0])) + "-" + convertdate(str(end2[1])) + "-" + convertdate(str(end2[2]))

start3 = find_sunday(end2[0], end2[1], end2[2])
end3 = find_saturday(start3[0], start3[1], start3[2])

startdate3 = convertdate(str(start3[0])) + "-" + convertdate(str(start3[1])) + "-" + convertdate(str(start3[2]))
enddate3 = convertdate(str(end3[0])) + "-" + convertdate(str(end3[1])) + "-" + convertdate(str(end3[2]))

sql = "select STUDY_A                STUDY_DATE between %s and %s) a
res = []
count = 0
for item in result:
    res.append(item)
    count+=1
d = dict()
d['target'] = count
# TODO implement
return {
    'statusCode': 200,
    'body': json.dumps(d)
}
```

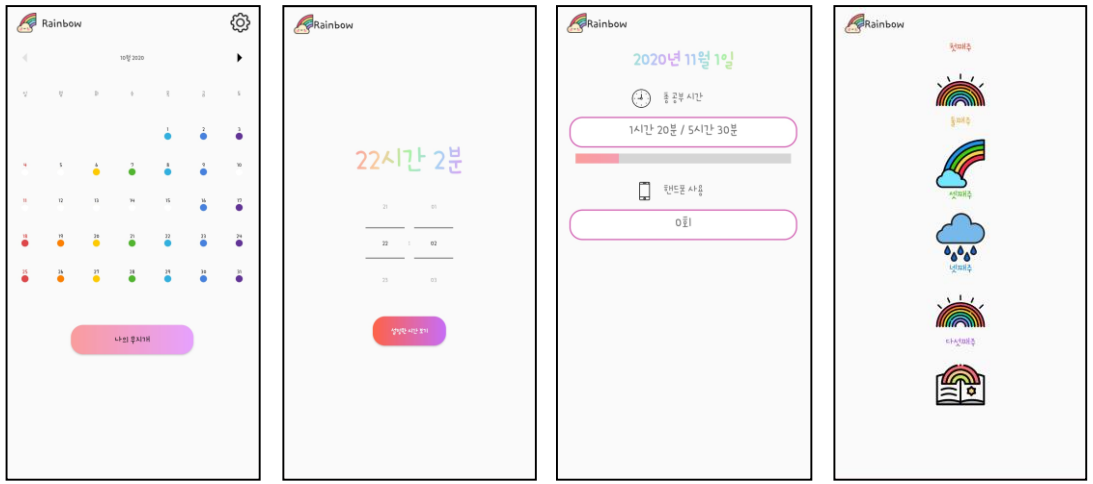
```
start2 = find_sunday(end1[0], end1[1], end1[2])
end2 = find_saturday(start2[0], start2[1], start2[2])

startdate2 = convertdate(str(start2[0])) + "-" + convertdate(str(start2[1])) + "-" + convertdate(str(start2[2]))
enddate2 = convertdate(str(end2[0])) + "-" + convertdate(str(end2[1])) + "-" + convertdate(str(end2[2]))
```

Part 6,

Result

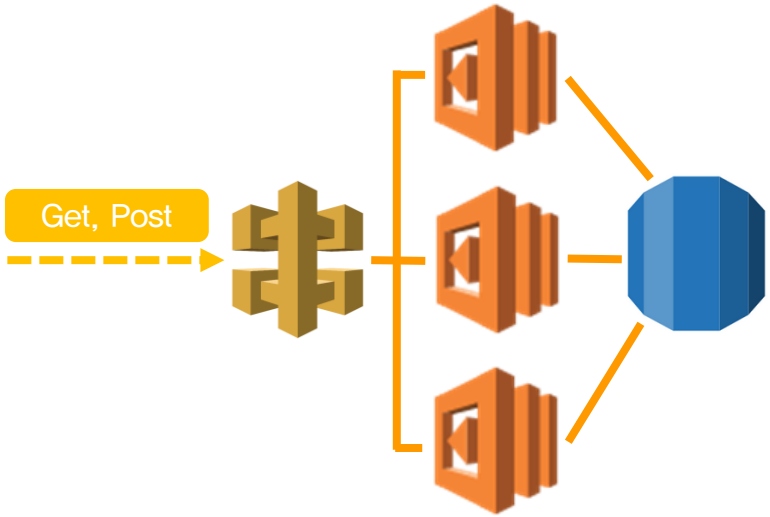
Rpi | Cloud | App

Result: Application



Application

Get, Post



AWS Cloud

Part 6,

Result

Rpi | Cloud | App

Result: Application

```
public interface RainbowAPI {

    @GET("goal")
    Call<PostItem> getGoalTime(@Query("target") String target);

    @GET("time")
    Call<PostItem> getStudyTime(@Query("target") String target);

    @GET("interrupt")
    Call<PostItem> getInterruptTime(@Query("target") String target);

    @GET("achievement")
    Call<PostItemStringList> getAchievementList(@Query("achieved") int target0, @Query("start") String target1, @Query("end") String target2);

    //@FormUrlEncoded
    @POST("goal")
    //Call<PostItemGoal> postGoal(@Field("date") String date, @Field("time") int time);
    Call<PostItemGoal> postGoal(@Body PostGoal newgoal);

    @GET("estimation")
    Call<PostItemEstimation> getInterruptTime(@Query("start") String target1, @Query("end") String target2);
}
```

쿼리 파라미터로 데이터 전송

HTTP method와 resource 명시

Result

Result: Application

```
package com.example.rainbow;

import com.google.gson.annotations.SerializedName;

public class PostGoal {

    @SerializedName("date") private String date;
    @SerializedName("time")private int time;

    public String getDate() { return date; }
    public int getTime() { return time; }

    public void setDate(String date) { this.date = date; }
    public void setTime(int time) { this.time = time; }

}
```

{ "Body" : { 'date' : string
'time' : integer } } 데이터를 클래스 변수 형태로 매핑

Part 6,

Result

Rpi | Cloud | App

Result: Application

```
Retrofit mRetrofit = new Retrofit.Builder()
        .baseUrl("https://r89kbtj8x9.execute-api.us-east-1.amazonaws.com/last/")
        .addConverterFactory(GsonConverterFactory.create())
        .build();
RainbowAPI mRetrofitAPI = mRetrofit.create(RainbowAP
```

레트로핏 사용을 위한 선언

```
Call<PostItemStringList> mCallMoviewList = mRetrofitAPI.getAchievementList(0,param[0],param[1]);
mCallMoviewList.enqueue(new Callback<PostItemStringList>() {
    @Override
    public void onResponse(Call<PostItemStringList> call,
        PostItemStringList result = response.body();
        new ApiSimulator(result.getLanget()).executeOnExecutor(Executors.newSingleThreadExecutor());
    }
    @Override
    public void onFailure(Call<PostItemStringList> call, Throwable t) {
        t.printStackTrace();
    }
});
```

앞서 선언한 getAchievementList를 통해
세 가지 데이터를 API gateway로 보냄

json body를 앞서 정의한 클래스 형태로 받아옴

Part 7,

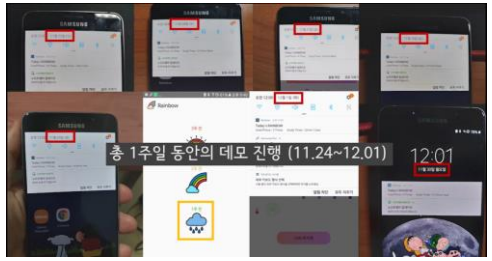
Demo

Part 7,

Demo

Demo Step

1. RPi wifi 연결 및 센서 설치
2. RPi 바이너리 파일 실행
3. App에서 목표 시간 설정
4. RPi가 설치된 책상 앞에서 공부 시작
5. 핸드폰으로 인터넷 접속 및 공부 시간 중간에 일어남
6. Buzzer 및 FND 센서 작동 확인
7. App에서 실제 공부 시간과 핸드폰 사용 횟수 확인
8. 자정에 푸시 알림을 통해 공부 시간 및 핸드폰 사용 횟수 확인
9. 해당 과정을 1주일간 반복



Demo

Part 7,

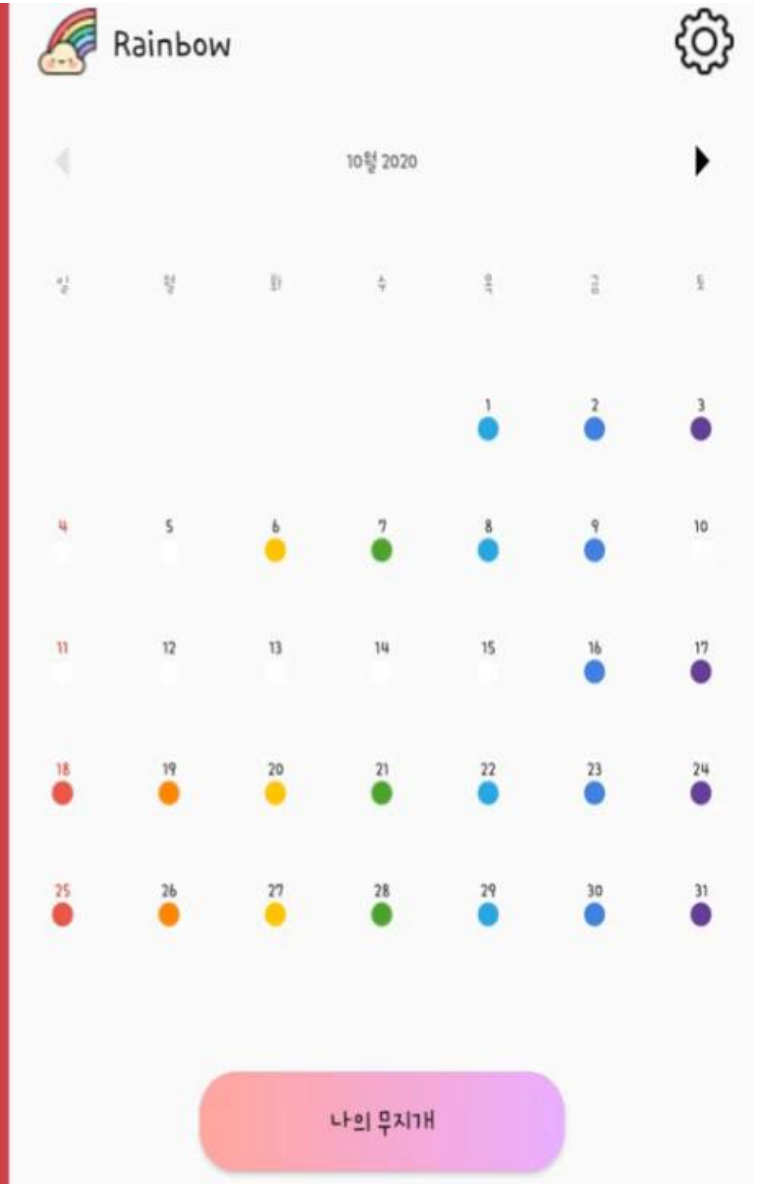
Demo

Be Attention

B조(집중해조)

강세희
김성수
서예진
이진아
최용욱

https://github.com/KHU-Rainbow



Part 8,

Division and Assignment of work

Part 8,

Division and Assignment of work

Division of work

work

IoT Device

공부 시간 측정 및 데이터 전송 자동화

Cloud

데이터베이스, 서버리스 컴퓨팅

Wireless Network Adapter

핸드폰 패킷 캡쳐

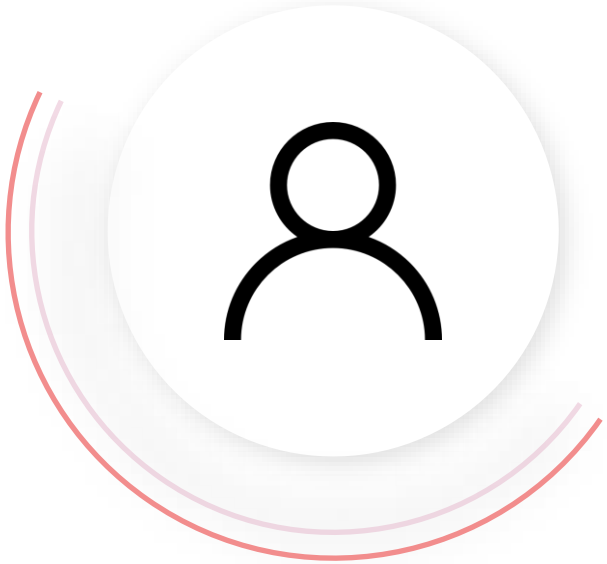
APP

AWS 데이터 시각화

Part 8,

Division and Assignment of work

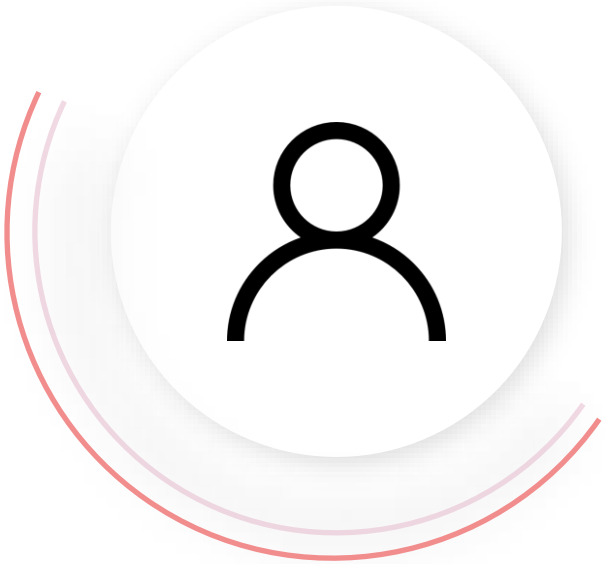
Assignment of work



김성수

담당 역할

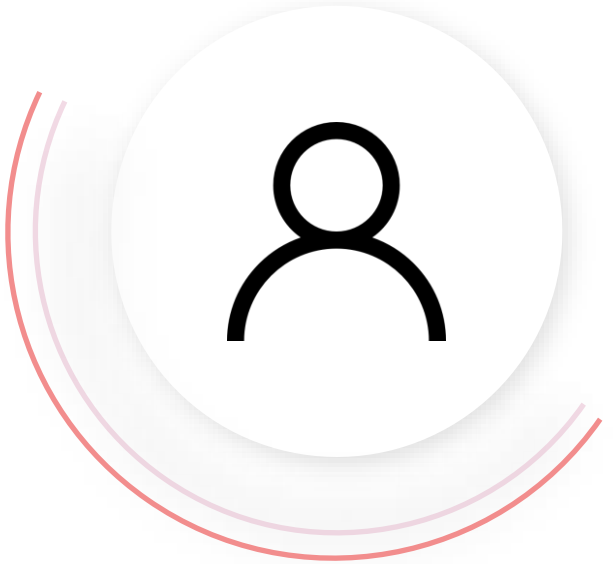
1. Rpi 개발 환경 구축
2. Rpi ↔ 센서: 초음파, FND, Buzzer
3. RPi ↔ AWS 인터페이스 구축
4. RDS, Lambda 초기 환경 구축
5. Android App Error 수정



서예진

담당 역할

1. 네트워크 패킷 구조체 작성
2. 네트워크 패킷 캡처모듈 개발
3. Firebase 푸시 알림



최용욱

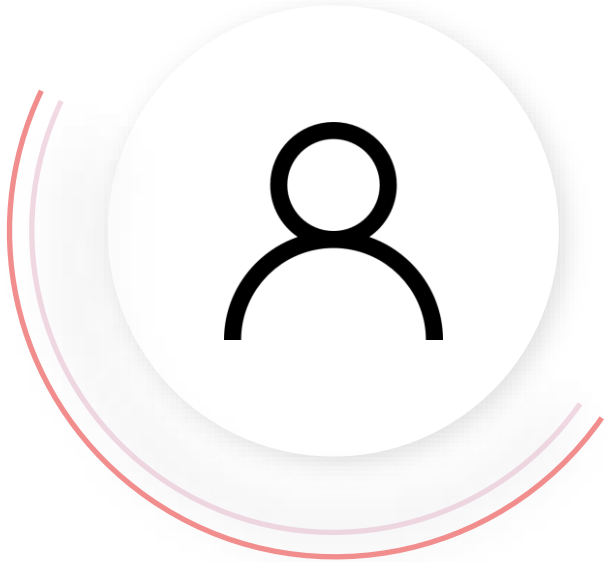
담당 역할

1. Server 구축
2. 데이터베이스 구축
3. Lambda 구축
4. Api Gateway 구축

Part 8,

Division and Assignment of work

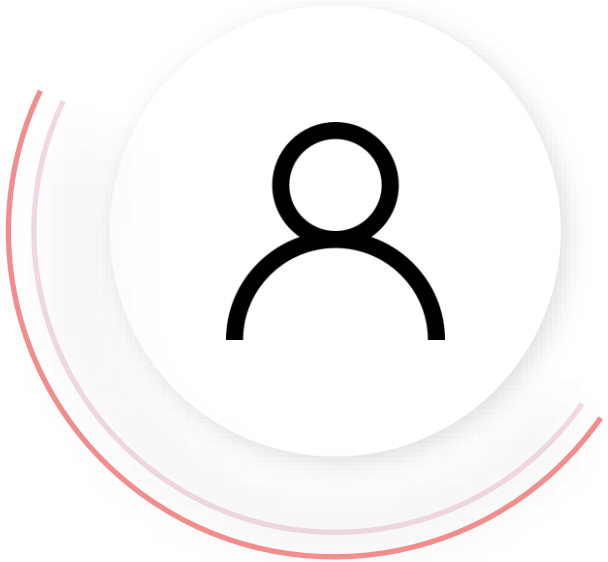
Assignment of work



강세희

담당 역할

1. 달력 페이지 기능 구성
2. Application SQLite 내장 DB 구축
3. AWS Cloud ↔ Application 연동



이진아

담당 역할

1. 일간 현황 페이지 기능 구성
2. 시간 설정 페이지 기능 구성
3. 주간 평가 페이지 기능 구성
4. 어플 디자인

Thank You