

SW개발/HW제작 설계서

프로젝트 명 : [20_HF001] IoT를 이용한 차량정보 전송 및 활용

2020. 07. 30

(CAN) – 정홍석, 강혜인, 우창민

Mentor 최무석

| 시장/기술 동향 분석

<표1> EU와 한국 서비스 시장의 비교

단위 : 점/100.0점

EU		MPI		한국	CMPI	
		점수	순위		점수	순위
서비스 시장 평균(31개 시장)		75.6		서비스 시장 평균 (29개 시장)	73.7	
Personal care services	이미용서비스	82.9	1	이미용서비스-미용실	75.2	2
Culture and entertainment	문화/오락	82.2	2	영화관람서비스	75.2	2
Commercial sport services	스포츠시설 이용	81.2	3	스포츠시설이용	74.1	13
Holiday accommodation	숙박서비스	81.0	4	숙박시설-펜션,콘도	74.4	10
Vehicle maintenance and repair	자동차수리서비스	75.1	15	자동차수리서비스	71.1	29

- 한국자동차 기술신문의 자동차 서비스시장과 변화대응에 따르면 한국의 자동차 수리서비스의 CMPI점수는 71.1이고 29위로 CMPI의 평균값인 73.7에도 미치지 못함.
- 이는 한국 뿐만이 아니라 유럽도 같은 현상을 보이며, EU 자동차수리서비스가 서비스분야 15위로 중위권 평가를 받고 있음.
- 따라서, 자동차 수리 서비스의 문제점을 느끼고 있는 것은 한국뿐 만이 아니라는 걸 알 수 있음.

| 시장/기술 동향 분석

<표2> 소비자 피해 발생 원인과 소비자지향성(요약)

단위 : 건

원인	소비자지향성	총합계
품질 (1,043)	만족도	1,043
부당행위 (274)	신뢰성	188
	가격	3
	소비자문제	83
의무 (62)	사업자선택가능성	62
정보 (42)	신뢰성	8
	가격	32
	소비자문제	2
안전	소비자문제	1
총 합계		1,422

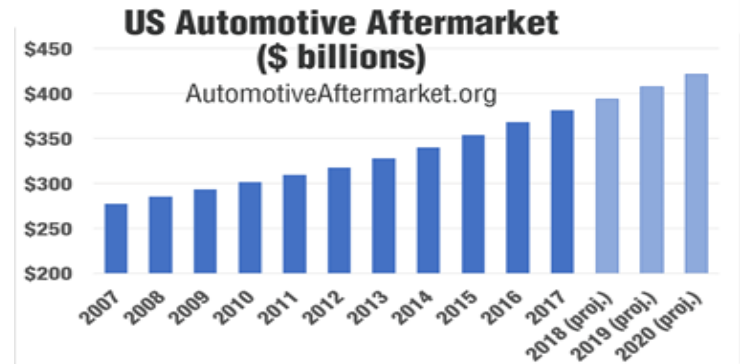
- 2015년 한국소비원에 접수된 자동차 서비스 불만신고들을 정리한 표를 보면 불만의 86.6%는 사업자가 자신의 일을 제대로 하지 않았기 때문에 생긴 것으로 나타남.
- 이는 자동차 서비스 센터에 대한 소비자들의 신뢰성이 떨어져 있음을 나타냄.
- 따라서, 자동차 수리 서비스의 문제점을 해결하기 위해 소비자들은 자신들의 차의 정보에 관심이 증대될 것 이다.

시장/기술 동향 분석

<표3>



<표4>



- 해외에선 애프터 마켓이 활성화 되어 소비자가 직접 부품을 구매하여 자동차를 관리하는 추세.
- 국내 자동차 애프터 마켓 시장의 규모는 이미 87조원대를 기록했으며, 현재 100조원대를 뛰어넘음.
- 미국의 애프터 마켓 시장은 2017년에 약 414조원대를 기록했으며, 2020년에는 약 473조의 크기의 시장을 가질 것이라고 예상됨

따라서, 저희 작품은 **타이어의 교체주기와 자동차의 부품시기를 예측하고 알려주는 기능**을 가지고 있어 **자동차 애프터 마켓을 이용하여 직접 수리하는 소비자들에게 필수적인 장비**가 될 것이라고 생각됨.

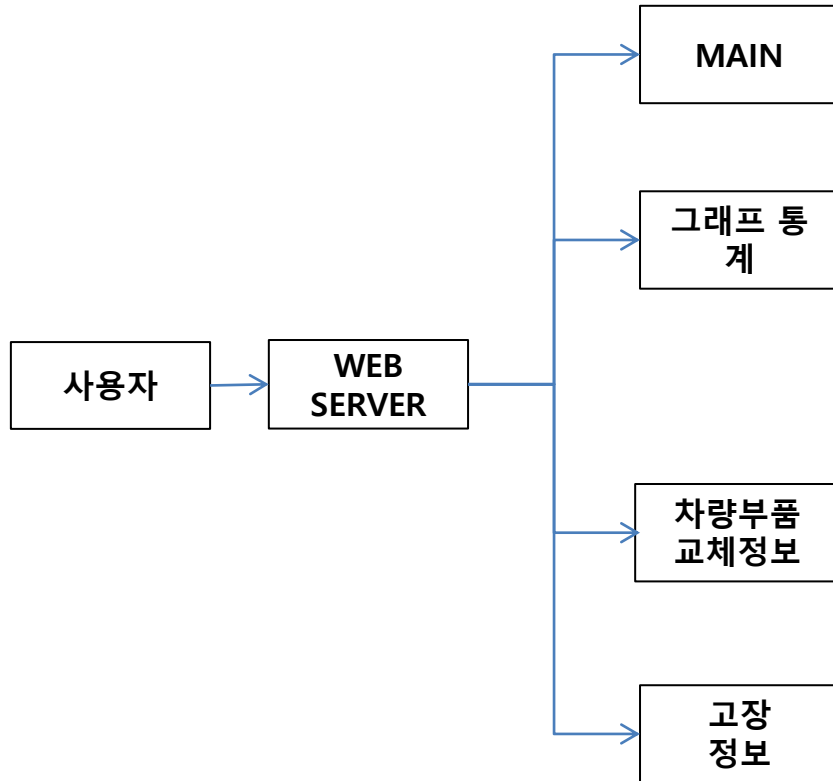
| 요구사항 정의서_1

요구사항 ID	요구사항명	기능 ID	기능명	세부사항	예외사항
A01	차량 데이터 값	A01_B01	RPM	RPM 값 표현	X
		A01_B02	Speed	Speed 값 표현	X
		A01_B03	ODO	ODO 값 표현	X
		A01_B04	Engine Temperature	엔진 냉각수 온도 값 표현	X
A02	차량 데이터 그래프	A02_B01	RPM	RPM 그래프 표현	X
		A02_B02	Speed	Speed 그래프 표현	X
		A02_B03	ODO	ODO 그래프 표현	X
		A02_B04	Engine Temperature	엔진 냉각수 온도 그래프 표현	X
A03	차량 부품 교체 정보	A03_B01 ~ A03_B11	엔진오일, 오토 미션오일, 파워 오일 등	교체 장비명과 교체 위험도를 표현	X
A04	고장 정보	A04_B01 ~ A04_B04	파워트레인(P) 통신(U) 샤시 시스템(C) 바디 시스템(B)	고장코드, 고장 위험도, 고장 내용 상세, 해결방법 표현	X

| 요구사항 정의서_2

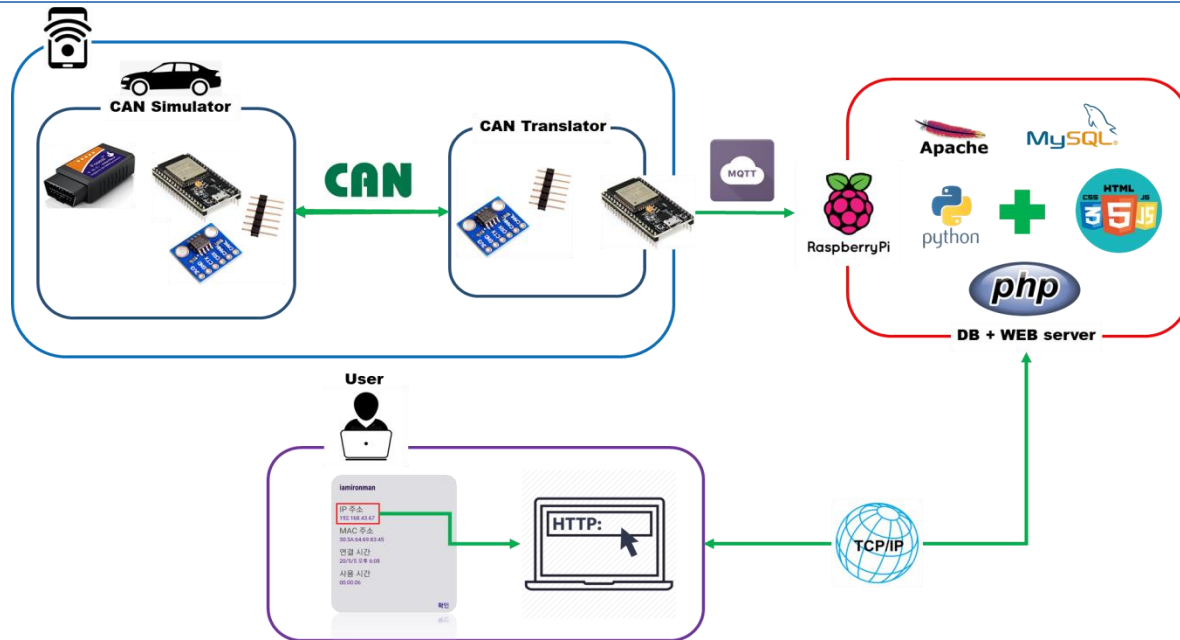
요구사항 ID	요구사항명	기능 ID	기능명	세부사항	예외사항
A05	CAN/ UDS 통신	A05_B01	SID	차량 데이터를 받기 위한 service ID	X
		A05_B02	PID	차량 데이터를 받기 위한 parameter ID	X
A06	인터넷 통신	A06_B01	HTTP	WIFI 서버와 데이터 교환	X
		A06_B02	PHP	WIFI 서버와 데이터 교환	X
		A06_B03	MQTT	WIFI 서버와 데이터 교환	X

| 서비스 구성도 - 서비스 시나리오



- Main
 - RPM, Speed, ODO, Engine Temperature
- 그래프 통계
 - RPM, Speed, ODO, Engine Temperature
- 차량 부품 교체 정보
 - 엔진오일, 와이퍼, 냉각수 등
- 고장정보
 - 고장코드, 고장위험도, 고장 내용, 해결방법

| 서비스 구성도 - 서비스 시나리오



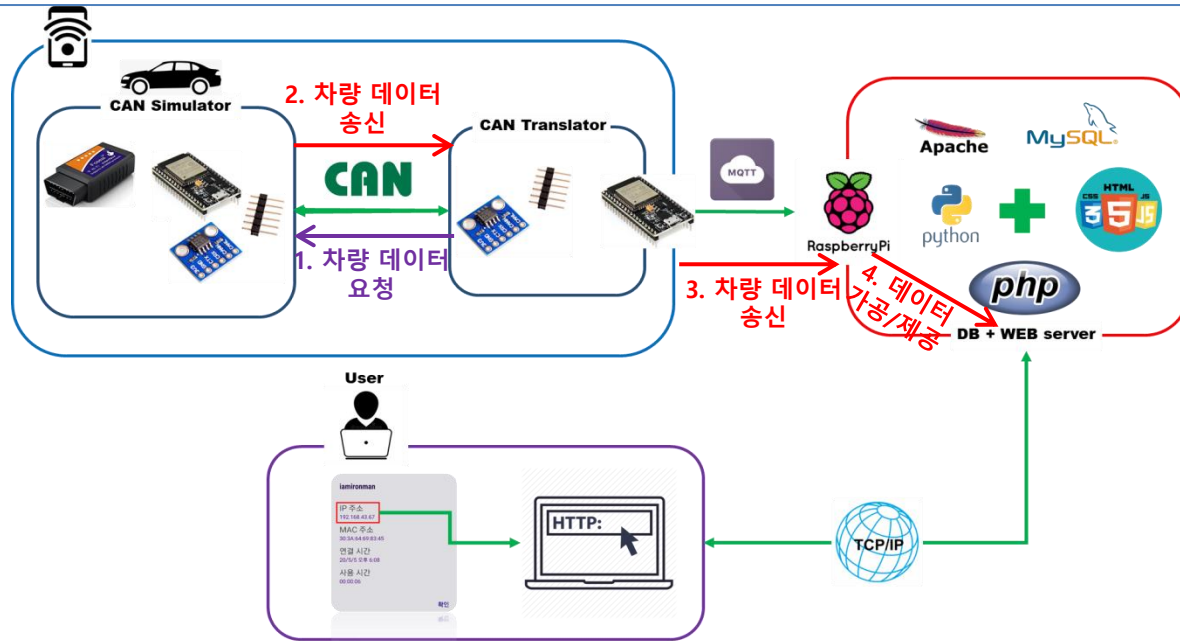
Back-end

- CAN 통신: 차량에서 데이터 수신
- TCP/IP 통신: 수신한 데이터 DB에 송신 및 저장 후 가공할 데이터는 파이썬을 통해 가공하고 일반 데이터는 그대로 WEBSERVER에 송신

Front-end

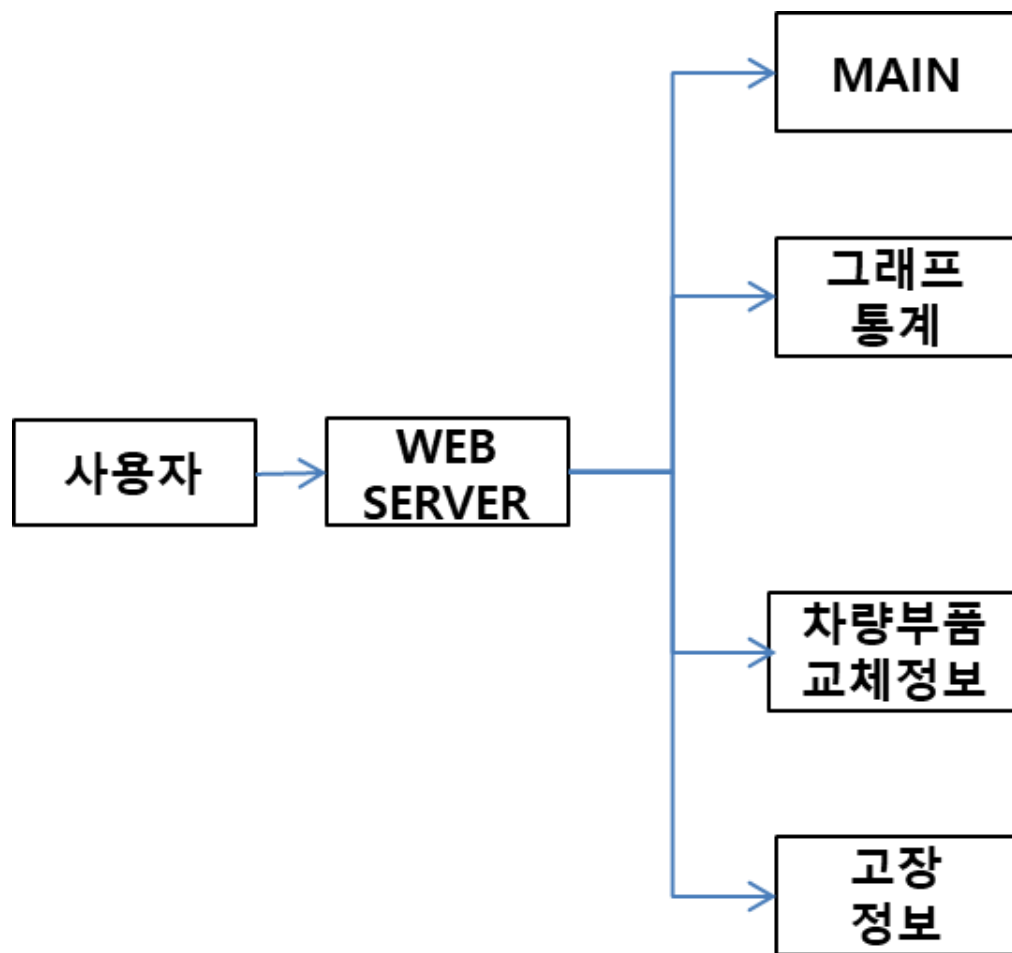
- 초기 데이터 요청: 부품 교체 시기
- 데이터 전달
- 데이터를 그래프화해서 전달

| 서비스 구성도 - 서비스 시나리오

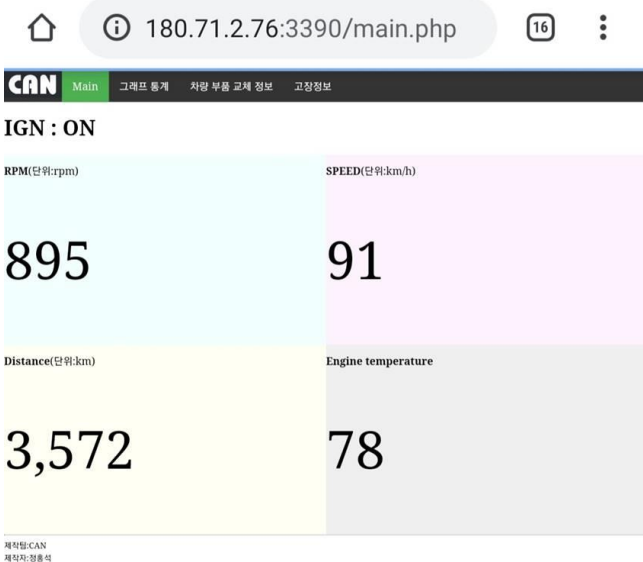


1. 차량 데이터 요청
2. 차량 데이터 송신(Simulator-> Translator)
3. 차량데이터 송신(Translator->DB)
4. 데이터 가공/제공(파이썬을 통해 데이터 가공/웹 서버로 가공한 데이터 제공)

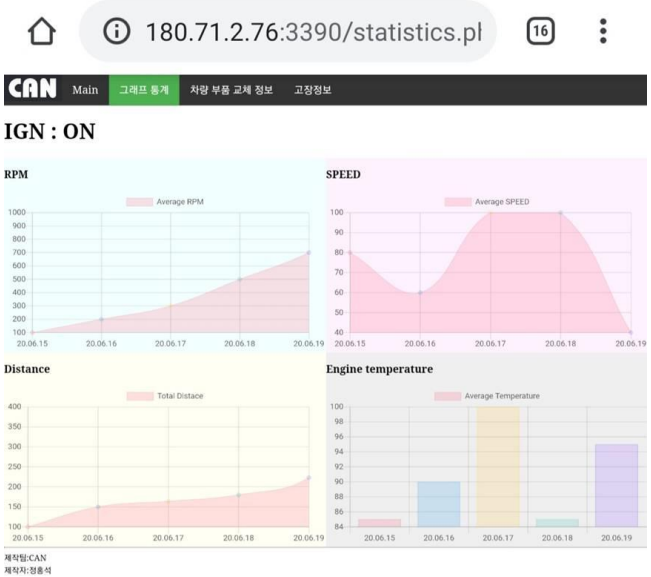
| 메뉴 구성도



| 화면 설계서

 <p>The screenshot shows a web browser at 180.71.2.76:3390/main.php. The interface has a header with 'CAN Main' and navigation links. The main content area displays four data points: RPM (895), SPEED (91), Distance (3,572), and Engine temperature (78). The status 'IGN : ON' is shown at the top left of the data area. The footer indicates '제작팀: CAN' and '제작자: 정종석'.</p>	기능 명	Main
	기능설명	웹의 첫 화면, 데이터 수치를 볼 수 있음
	처리내용	-RPM, Speed, ODO, Engine Temperature 값들을 db에서 받아와서 화면에 출력
	비고	없음
	요구사항 명	데이터 값 표현

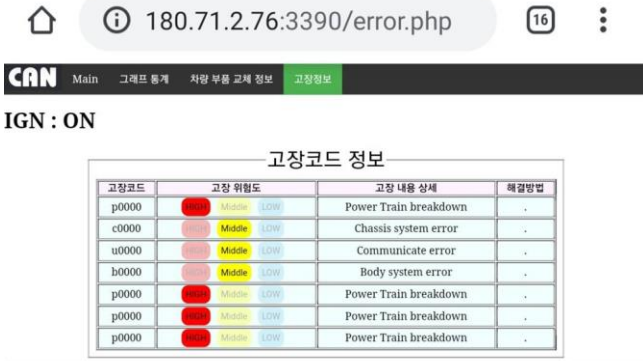
| 화면 설계서

 <p>The screenshot shows a web browser at 180.71.2.76:3390/statistics.pl. The interface has a navigation bar with 'Main', '그래프 통계' (selected), '차량 부품 교체 정보', and '고장정보'. Below the bar, it says 'IGN : ON'. There are four graphs: 'RPM' (Average RPM), 'SPEED' (Average SPEED), 'Distance' (Total Distance), and 'Engine temperature' (Average Temperature). Each graph shows data from 20.06.15 to 20.06.19.</p>	<p>기능 명</p>	<p>그래프 통계</p>
	<p>기능설명</p>	<p>웹의 두 번째 화면, 데이터를 시각화한 그래프를 볼 수 있음</p>
	<p>처리내용</p>	<p>-RPM, Speed, ODO, Engine Temperature 값들을 DB에서 파이썬으로 가공한 후 받아와 그래프를 화면에 출력</p>
	<p>비고</p>	<p>없음</p>
	<p>요구사항 명</p>	<p>그래프 표현</p>

| 화면 설계서

 <p>제작팀: CAN 제작자: 임종석</p>	기능 명	차량 부품 교체 정보
	기능설명	웹의 세 번째 화면, 부품 교체 정보를 볼 수 있음
	처리내용	차량 부품 교체 장비명 - 엔진오일, 와이퍼, 냉각수 등 교체 위험도 - High, Middle, Low
	비고	없음
	요구사항 명	차량 부품 교체 정보 표현

| 화면 설계서

	기능 명	고장정보
	기능설명	웹의 네 번째 화면, 고장 코드를 볼 수 있음
	처리내용	고장코드 고장위험도: High, Middle, Low 고장 내용 해결방법
	비고	없음
	요구사항 명	데이터 값 표현

| 엔티티관계도 - ERD



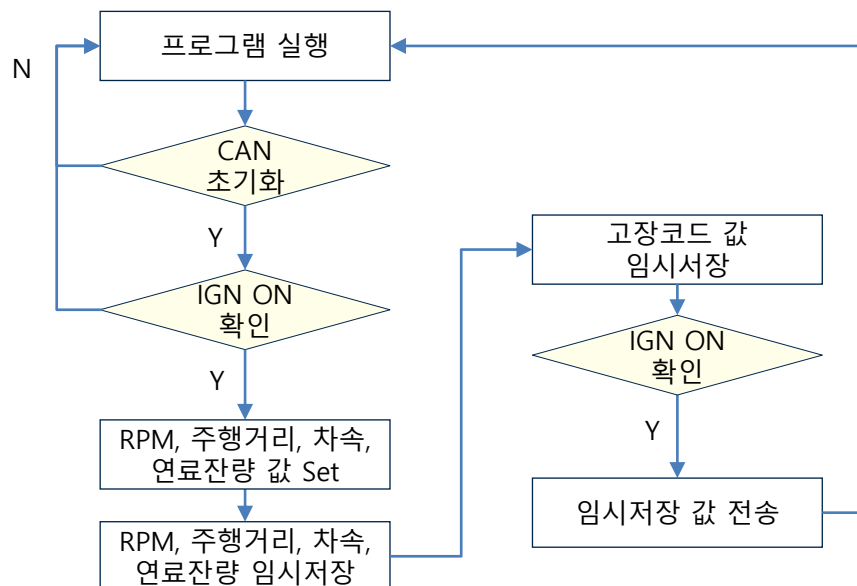
| 기능 처리도(기능 흐름도)

실무 산출물 형식

프로그램 ID	ESP32_1	프로그램 명	OBD2_ECU_Simulator.ino	작성일	2020.07.30	Page	1/8
개요	차량과 OBD 및 UDS 통신을 수행하여 실시간으로 차량 운행 데이터를 수집하고 DB에 전송하는 프로그램					작성자	정홍석

기능 흐름도

<차량 시뮬레이터>

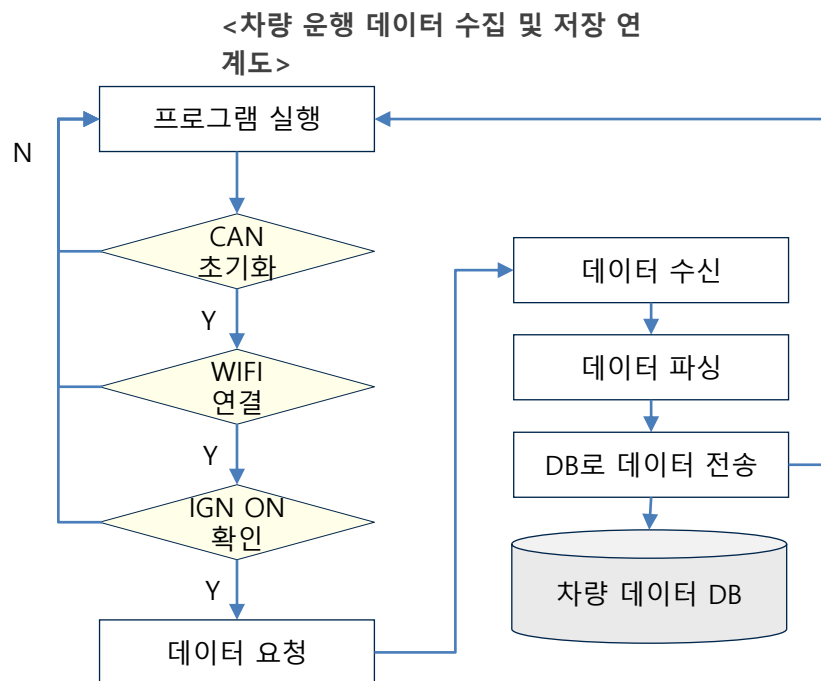


| 기능 처리도(기능 흐름도)

실무 산출물 형식

프로그램 ID	ESP32_2	프로그램 명	hanium_from_esp32_DB.ino	작성일	2020.07.30	Page	2/8
개요	차량과 OBD 및 UDS 통신을 수행하여 실시간으로 차량 운행 데이터를 수집하고 DB에 전송하는 프로그램					작성자	정홍석

기능 흐름도



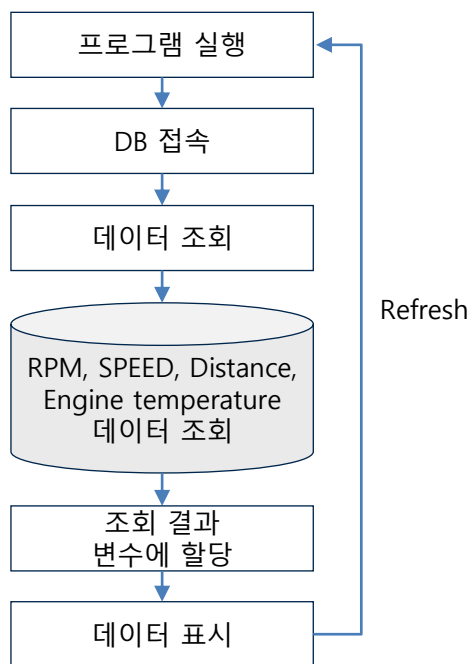
| 기능 처리도(기능 흐름도)

실무 산출물 형식

프로그램 ID	Rasp_server_1	프로그램 명	Main.php	작성일	2020.07.30	Page	3/8
개요	수집된 데이터를 DB에서 가져와 웹상에 표시해주는 프로그램					작성자	정홍석

기능 흐름도

<Main.php 기능 흐름도>



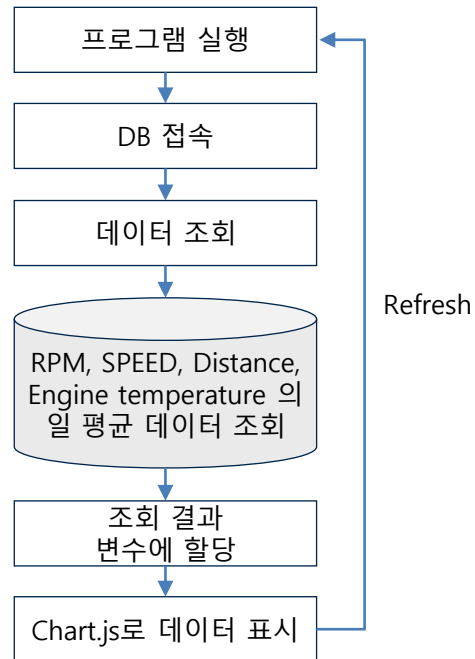
| 기능 처리도(기능 흐름도)

실무 산출물 형식

프로그램 ID	Rasp_server_2	프로그램 명	statistics.php	작성일	2020.07.30	Page	4/8
개요	일평균 데이터를 DB에서 가져와 웹상에 그래프로 표시해주는 프로그램					작성자	정홍석

기능 흐름도

<statistics.php 기능 흐름도>



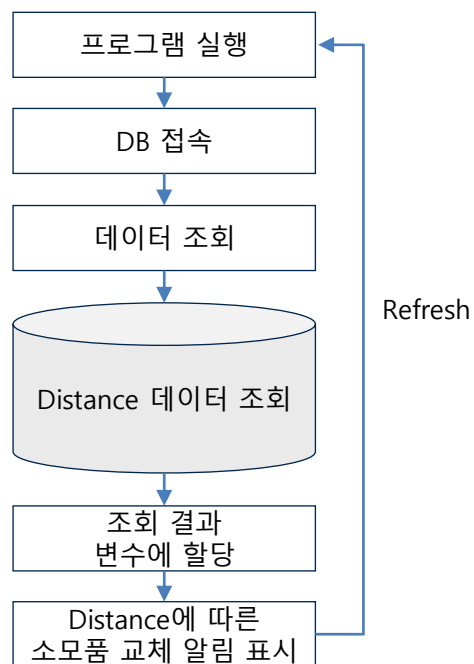
| 기능 처리도(기능 흐름도)

실무 산출물 형식

프로그램 ID	Rasp_server_3	프로그램 명	exchange.php	작성일	2020.07.30	Page	5/8
개요	차량의 총 운행 거리에 따른 소모품 교체 알림을 시각적으로 보여주기 위한 프로그램					작성자	정홍석

기능 흐름도

<exchange.php 기능 흐름도>



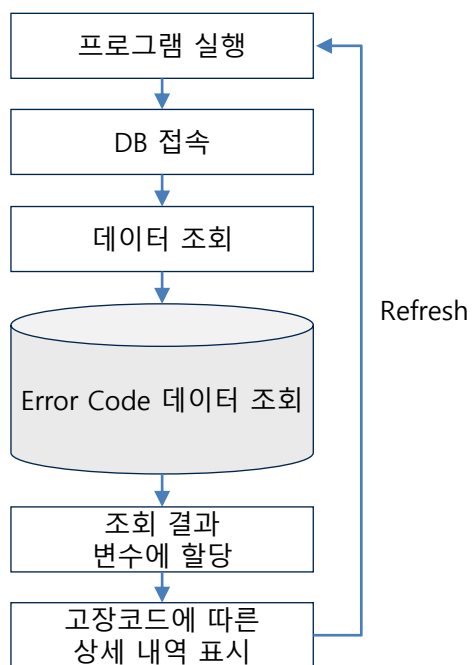
| 기능 처리도(기능 흐름도)

실무 산출물 형식

프로그램 ID	Rasp_server_4	프로그램 명	error.php	작성일	2020.07.30	Page	6/8
개요	UDS 프로토콜을 통해 DB에 저장된 고장코드 정보를 웹에 표시하는 프로그램					작성자	정홍석

기능 흐름도

<error.php 기능 흐름도>



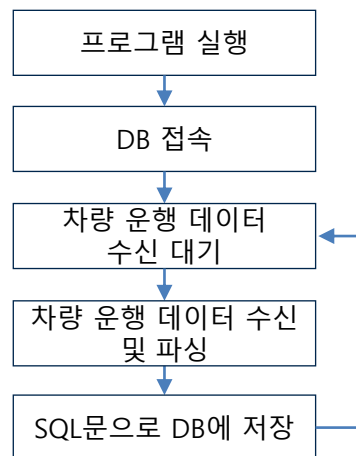
| 기능 처리도(기능 흐름도)

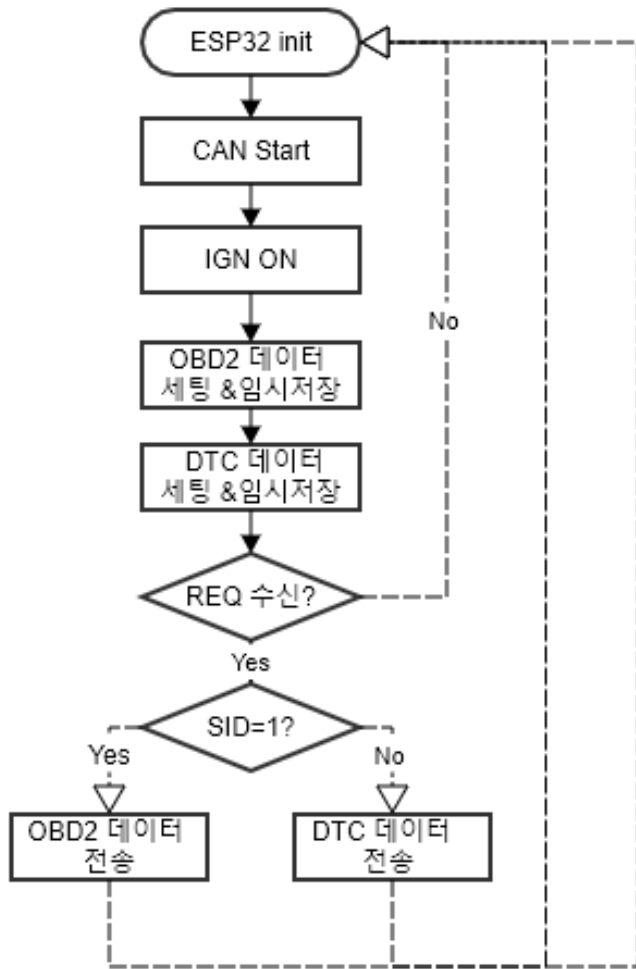
실무 산출물 형식

프로그램 ID	Rasp_server_5	프로그램 명	dbtest.php	작성일	2020.07.30	Page	7/8
개요	Esp32에서 전송되는 차량 운행 데이터를 DB에 저장					작성자	정홍석

기능 흐름도

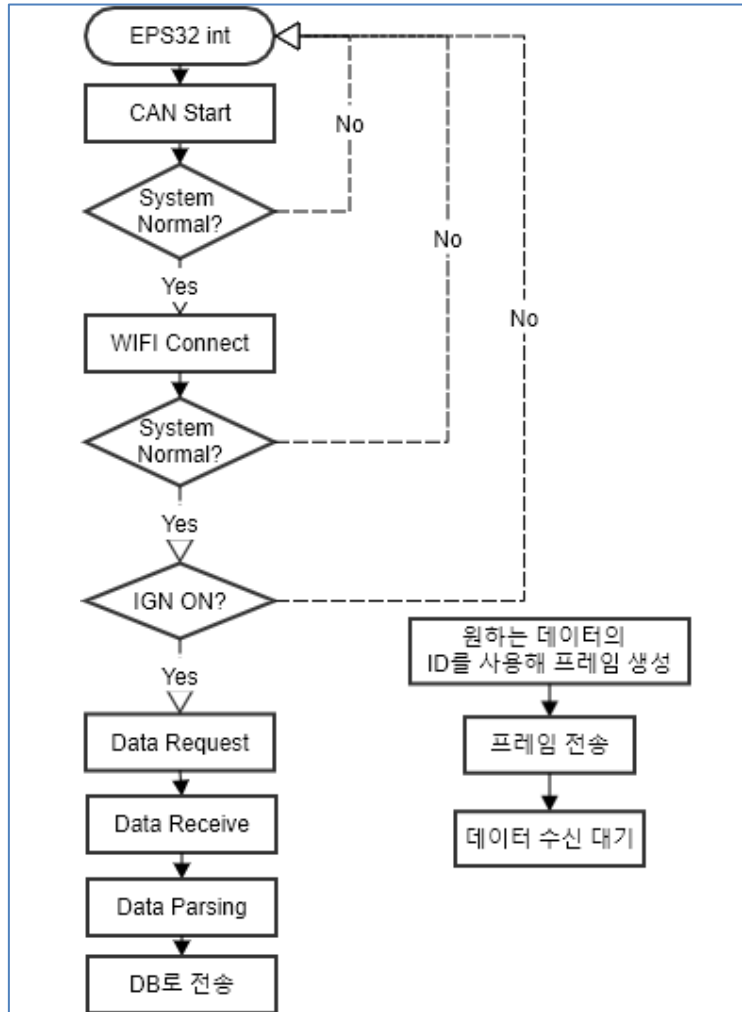
<dbtest.php 기능 흐름도>





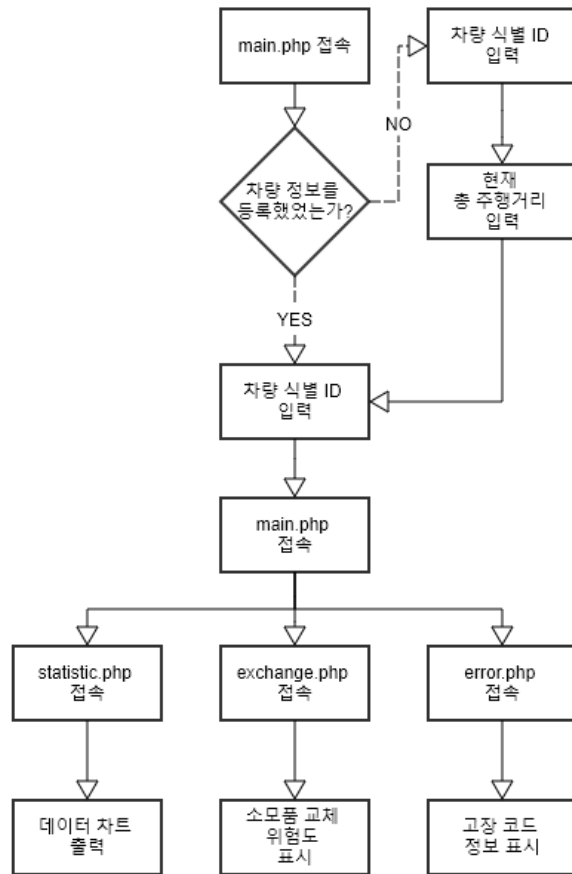
1. 값들을 초기화
2. CAN 통신연결
3. 엔진이 켜져 있는지 체크
4. 가변저항을 통해 차량 obd 값 세팅&저장
5. 스위치를 통해 DTC 값 세팅&저장
6. Translator 측으로부터 요청이 왔는지 체크하고 안 왔으면 1번으로 돌아가고 왔으면 다음
7. SID= 1(Show Current Data)
SID= 3(Read DTC)
8. SID=1이면 OBD2 값들을 전송
9. SID=3이면 DTC 값들을 전송

| 알고리즘 명세서_Translator



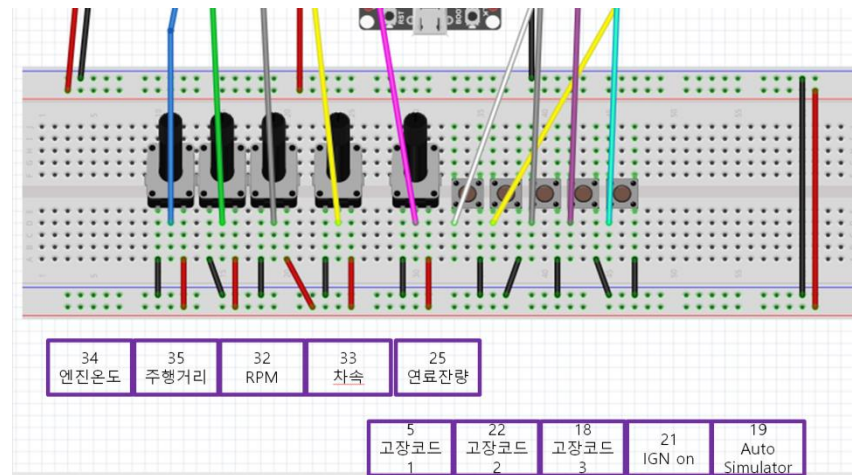
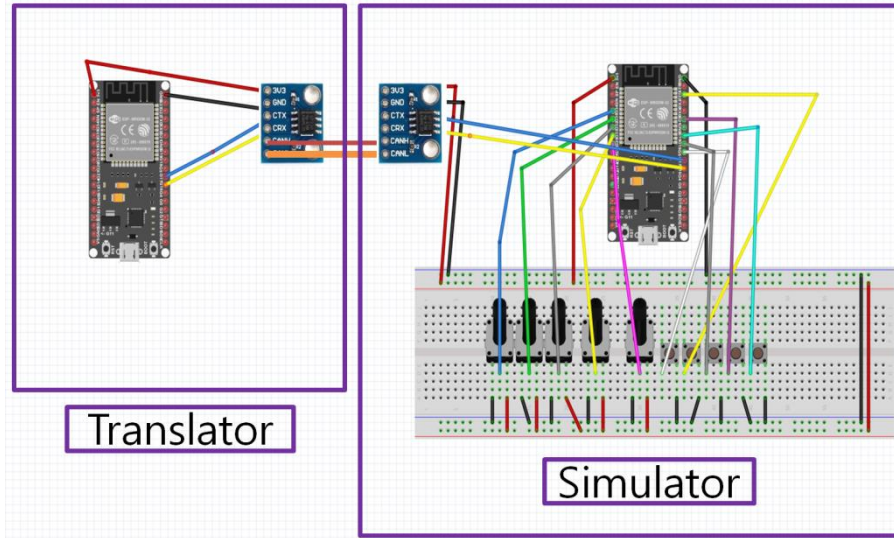
1. 값들을 초기화
2. CAN 통신이 가능한지 체크
3. 시스템 정상인지?
4. WIFI 연결
5. 시스템 정상인지?
6. 엔진이 켜져 있는지?
7. DATA 요청-1)원하는 데이터 ID 프레임화
2)프레임 전송
3)데이터 수신 대기
8. 데이터 수신
9. 데이터 파싱
10. DB로 전송

| 알고리즘 명세서_WEB



1. 첫 페이지인 main에 접속
2. 차량 정보 등록을 했었는지 체크
- 2-no. 차량 식별 ID 입력->현재 총 주행거리 입력
3. 차량 식별 ID 입력
4. 차량 식별 ID를 통해 사용자의 정보를 main에 실시간 출력
- 4-1. 통계 페이지 접속
- 5-1. 데이터 차트 출력
- 4-2. 소모품 교체 페이지 접속
- 5-2. 소모품 교체 위험도 표시
- 4-3. 고장 코드 페이지 접속
- 5-3. 고장 코드 정보 표시

| 하드웨어 설계도



| 프로그램 - 목록

기능 분류	기능번호	기능 명
Sav	Sav-01	Php에서 차량 운행 데이터 DB에 전송
	Sav-02	Python을 이용하여 가공된 데이터 DB에 전송
Cmc	Cmc-01	ESP32에서 OBD2, UDS를 통한 데이터 송수신
	Cmc-02	ESP32에서 WIFI 연결 및 데이터 전송
Calc	Calc-01	각 데이터 일 평균 계산
	Calc-02	급가속, 급감속 계산
Web	Web-01	데이터 웹에 표시
	Web-02	데이터를 그래프로 웹에 표시
	Web-03	차량 부품 교체 정보 표시
	Web-04	차량 고장 정보 표시

| 테이블 정의서 - ERD



| 테이블 정의서 - ERD

Env_data

테이블 설명: Car enviroment data using Can Protocol...

컬럼명	종류	Null	기본값	링크 대상	설명	MIME
id (기본)	int(32)	아니오				
DT	timestamp	아니오	current_timestamp()			
IGN	int(2)	예	NULL			
RPM	int(38)	예	NULL			
D	int(38)	예	NULL			
V	int(38)	예	NULL			
Eng_temp	int(38)	예	NULL			
Fuel_level	int(38)	예	NULL			

인덱스

키 이름	종류	고유값	압축됨	컬럼명	관계성	데이터정렬방식	Null	설명
PRIMARY	BTREE	예	아니오	id	0	A	아니오	

| 테이블 정의서 - ERD

avg_data

테이블 설명: save calculated data...

컬럼명	종류	Null	기본값	링크 대상	설명	MIME
id (기본)	int(32)	아니오				
Env_data_id	int(32)	예	NULL			
DT	date	아니오	current_timestamp()			
avg_rpm	int(38)	예	NULL			
avg_speed	int(38)	예	NULL			
avg_Distance	int(38)	예	NULL			
avg_Eng_temp	int(38)	예	NULL			
avg_Fuel	int(38)	예	NULL			

인덱스

키 이름	종류	고유값	압축됨	컬럼명	관계성	데이터정렬방식	Null	설명
PRIMARY	BTREE	예	아니오	id	0	A	아니오	

| 테이블 정의서 - ERD

error_code

테이블 설명: error_code and resolve

컬럼명	종류	Null	기본값	링크 대상	설명	MIME
id (기본)	int(32)	아니오				
Env_data_id	int(32)	예	NULL			
DT	timestamp	예	current_timestamp()			
code	varchar(32)	예	NULL			
Danger	int(3)	예	NULL			
Detail	varchar(32)	예	NULL			
res	varchar(32)	예	NULL			

인덱스

키 이름	종류	고유값	압축됨	컬럼명	관계성	데이터정렬방식	Null	설명
PRIMARY	BTREE	예	아니오	id	4	A	아니오	

| 핵심소스코드(1)

```
#include <esp32_can.h>
#include <iso-tp-esp32.h>
#include <uds-esp32.h>

ESP32_CAN CAN0;
IsoTp isotp(&CAN0);
UDS uds(&isotp);

//가변저항
const int CoolantTemp_pin = 34;
const int Odometer_pin = 35;
const int EngineRpm_pin = 32;
const int VehicleSpeed_pin = 33;
const int EngineFuelRate_pin = 25;

//스위치
const int Error_Code1_pin = 5;
const int Error_Code2_pin = 22;
const int Error_Code3_pin = 18;
const int IGN_ON_pin = 21;
const int Auto_Simulator_pin = 19;

struct Session_t session;
//8비트로 잘라서 배열에 넣어주는 함수
void hex_converter_8(int num, uint8_t bitarray[], int digit); //들어온 값 , 값을 분해해서 넣을 배열, 배열이 몇 digit으로 구성되어 있는지
void RandomNumGenerator(int Max, uint8_t bitarray[], int digit );
//uint32_t map_32(uint32_t x, uint32_t in_min, uint32_t in_max, uint32_t out_min, uint32_t out_max);

uint8_t pidSupport0x01To0x20[] = {0x08, 0x18, 0x00, 0x00};
uint8_t pidSupport0x81To0xA0[] = {0x00, 0x00, 0x00, 0x08};
uint8_t pidSupport0xA1To0xC0[] = {0x04, 0x00, 0x00, 0x00};
uint8_t pid0x05CoolantTemp[] = {0x78}; // 0x78 = 120 => 120 - 40 = 80 C
uint8_t pid0x0CEngineRpm[] = {0x1F, 0x40}; // 0x1F 40 = 8000 => 8000 / 4 = 2000 rpm
uint8_t pid0x0DVehicleSpeed[] = {0x64}; // 0x64 = 100 Km/h
uint8_t pid0x9DEngineFuelRate[] = {0x3C}; //0x3C=60L
uint8_t pid0xA6Odometer[] = {0x00, 0x01, 0x86, 0xA0}; //0x00 01 86 A0=100000km

//
//      min      MAX      Formula
// temperature -40      215      A-40
// RPM          0      16383.75      (256*A+B)/4      -> 4digit
// Speed        0      255      A
// Fuel          x      x      x
// Odometer     0      5000000      (A(2^24)+B(2^16)+C(2^8)+D)/10 ->8digit

#define NUM_PIDS 8
struct {
    uint8_t pid;
    uint8_t* Data;
} responseData[NUM_PIDS] = {{0x00, pidSupport0x01To0x20},
    {0x05, pid0x05CoolantTemp},
    {0x0C, pid0x0CEngineRpm},
    {0x0D, pid0x0DVehicleSpeed},
    {0x80, pidSupport0x81To0xA0},
    {0x9D, pid0x9DEngineFuelRate},
    {0xA0, pidSupport0xA1To0xC0},
    {0xA6, pid0xA6Odometer}
};
```


| 핵심소스코드(2)

```
void loop()
{
    uint8_t rxData[LEN_DATA];
    uint8_t txData[LEN_DATA];
    struct Session_t diag;
    uint16_t retval = 0;
    uint8_t pid;

    Serial.println(F("OBD2 ECU server: "));
    diag.tx_id = 0x7E8;
    diag.rx_id = 0x7E0;
    diag.Data = rxData;

    //IGNON ON됨으로 자동차 정보 값을 배열에 저장
    int reading_Odometer = analogRead(Odometer_pin);
    uint32_t Odometer = map(reading_Odometer, 0, 4095, 0, 50000000);
    Serial.println("Odometer");
    hex_converter_8(Odometer, pid0xA6Odometer , 8);
    Serial.println();

    // TODO : Need to consider fomular, min, max of PDI tabale.
    //temperature=A-40
    int reading_CoolantTemp = analogRead(CoolantTemp_pin);
    int CoolantTemp = map(reading_CoolantTemp, 0, 4095, 0, 255);
    // Serial.println("Engine Temperature");
    // hex_converter_8(CoolantTemp, pid0x05CoolantTemp , 2);
    // Serial.println();

    //RPM =A-40
    int reading_EngineRpm = analogRead(EngineRpm_pin);
    uint32_t EngineRpm = map_32(reading_EngineRpm, 0, 4095, 0, 65535);
    // Serial.println("Engine RPM");
    // hex_converter_8(EngineRpm, pid0x0CEngineRpm , 4);
    // Serial.println();

    int reading_VehicleSpeed = analogRead(VehicleSpeed_pin);
    int VehicleSpeed = map(reading_VehicleSpeed, 0, 4095, 0, 255);
    // Serial.println("Vehicle Speed");
    // hex_converter_8(VehicleSpeed, pid0x0DVehicleSpeed , 2);
    // Serial.println();

    int reading_EngineFuelRate = analogRead(EngineFuelRate_pin);
    int EngineFuelRate = map(reading_EngineFuelRate, 0, 4095, 0, 60);
    // Serial.println("Engine Fuel Rate");
    // hex_converter_8(EngineFuelRate, pid0x9DEngineFuelRate , 2);
    // Serial.println();

    int reading_Error_Code1 = digitalRead(Error_Code1_pin);
    int reading_Error_Code2 = digitalRead(Error_Code2_pin);
    int reading_Error_Code3 = digitalRead(Error_Code3_pin);
    int reading_Auto_Simulator = digitalRead(Auto_Simulator_pin);
}
```

| 핵심소스코드(3)

```

if (retval = uds.SessionServer(&diag))
{
    Serial.println(F("No OBD request from tool."));
}
else
{
    Serial.print(F("request SID: ")); Serial.println(diag.sid);
    pid = diag.Data[0];
    Serial.print(F("request PID: ")); Serial.println(pid);
    Serial.print(F("request data: "));
    for (uint8_t i = 1; i < diag.len; i++)
    {
        Serial.print(diag.Data[i]); Serial.print(F(" "));
    }
    Serial.println();

    memset(txData, 0, LEN_DATA);
    for (uint8_t i = 0; i < NUM_PIDS; i++)
    {
        if (responseData[i].pid == pid) // Find the required pid data
        {
            diag.sid = diag.sid + 0x40; // Add 0x40 to received sid
            txData[0] = pid;
            uint8_t lenPidValue = sizeof(responseData[i].Data);
            memcpy(txData + 1, responseData[i].Data, lenPidValue);
            diag.Data = txData;
            diag.len = lenPidValue + 1;
            retval = uds.serverResponse(&diag);
        }
    }
    delay(1000);
}

void hex_converter_8(uint32_t num, uint8_t bitarray[], int digit) { //들어온 값 , 값을 분해해서 넣을 배열, 배열이 8bit로 몇개 구성되어 있는지
    int pos = digit;
    int temp[digit] = { 0 }; // 16진수로 된 문자열을 임시저장할 배열 (순서 제대로임)
    uint32_t decimal = num;

    for(int i=0; i<pos;i++){
        if(decimal<16){
            temp[pos-i-1]=decimal;
        }
        temp[pos-i-1]=decimal%16;
        decimal=decimal/16;
    }
    Serial.print(" 0x");
    for (int j = 0; j < pos/ 2; j++) {
        bitarray[j] = temp[2 * j] * 16 + temp[2 * j + 1]; //int로 잘려져있는 값들을 8비트 배열에 차곡차곡 넣어줌
        Serial.print(bitarray[j], HEX); Serial.print(" ");
    }
}

```

| 참조- 개발 환경 및 설명

구분		항목	적용내역
SW 개발환경	CAN Simulator/ CAN Translator 개발	Arduino IDE (1.8.9)	차량 역할을 할 CAN Simulator와 차량에서 Data를 받아오는 CAN Translator 개발
	서버 애플리케이션 개발	PHP(5.3.3)	서버 관리자 웹 페이지 처리 모듈 작성
		MySQL(5.1.73)	차량 데이터를 저장, 관리하는 데이터베이스
		Apache(1.7.1)	관리자 웹 페이지를 구동하는 웹 서버
		서버 운영체제	리눅스 Rasbian OS
		Putty	Rasbian OS 원격 접속용 툴
H/W 구성장비	디바이스	스마트폰 (안드로이드, 아이폰)	사용자에게 서비스를 직접적으로 제공하는 End Device
		ESP 32	CAN Simulator, CAN Translator 제작에 필요한 MCU
	통신	WIFI	ESP32와 Raspberry Pi 4 통신
		무선 랜 어댑터	스마트폰과 웹서버 통신
	서버	Raspberry Pi 4	웹 호스팅 및 DB 정보를 변환해 다시 저장해주는 서버

Thank you

