□ 서식4_한이음 ICT멘토링 프로젝트 결과보고서

2020 한이음 ICT멘토링

프로젝트명	IoT를 이용한 차량정보 전송 및 활용

요 약 본

프로젝트 정보	
주제영역	■ 생활 □ 업무 ■ 공공/교통 □ 금융/핀테크 □ 의료 □ 교육 □ 유통/쇼핑 □ 엔터테인먼트
기술분야	■ IoT □ 모바일 □ 데스크톱 SW □ 인공지능 □ 보안 □ 가상현실 □ 빅데이터 □ 자동제어기술 □ 블록체인 □ 영상처리
달성성과	■ 논문게재 및 포스터발표 □ 앱등록 □ 프로그램등록 □ 특허 □ 기술이전 □ 실용화 ■ 공모전(2020 한이음 공모전) □ 기타( )
프로젝트명	IoT를 이용한 차량정보 전송 및 활용
프로젝트 소개	○ 차량상태(차속, 엔진속도, 엔진수온, 밧데리전압), 차량내 ECU의 진단 정보(DTC)를 인터넷을 통하여 전송함으로 원격지에서도 차량상태, 진단 정보를 확인하고 분석할 수 있는 기반기술을 개발하는 것
개발배경 및 필요성	○ 차량상태와 진단정보는 차량이 있는 장소에서 진단장비를 차량의 OBD 단자에 접속해야만 얻을 수 있으므로 차량운전자와 정비책임자가 모두 동일 시간, 장소에 있어야 함 ○ 이 기술은 차량상태와 진단정보를 원격으로 전송함으로서 시간과 공간의 제약을 해결하고 수집된 차량 데이터를 분석하여 부가서비스를 개발할 수 있음
프로젝트 주요기능	○ 차량정보 및 진단정보를 UDS통신규약을 사용해서 CAN통신을 통해 받아 Web을 통해서 확인 ○ 수집된 차량정보를 토대로 부품 교체시기를 알림 ○ 실시간 차량정보를 수집하여 운전 습관 개선 도움
작품의 기대효과 및 활용분야	○ 차량 데이터를 웹을 통하여 시각적으로 보여줌으로 데이터 접근성 및 활용 증대 ○ 차량 데이터를 DB를 통하여 체계적으로 관리할 수 있는 기반기술을 제공으로 차량운행, 정비 관리를 자동화 하여 차량수명 증가 및 관리비용 감소 ○ 차량 운행정보와 차량 정비 데이터를 DB에 저장함으로써 차량배치, 부품 교체시기 예측과 같은 부가서비스와 연계 가능

(본문) 프로젝트 결과보고서

Ⅰ. 프로젝트 개요

가. 프로젝트 소개

ㅇ 차량상태(차속, 엔진속도, 엔진수온, 밧데리전압), 차량내 ECU의 진단정보(DTC)를 인터넷을 통하여 전송함으로 원격지에서도 차량상태, 진단정보를 확인하고 분석할 수 있는 기반기술 개발하는 것

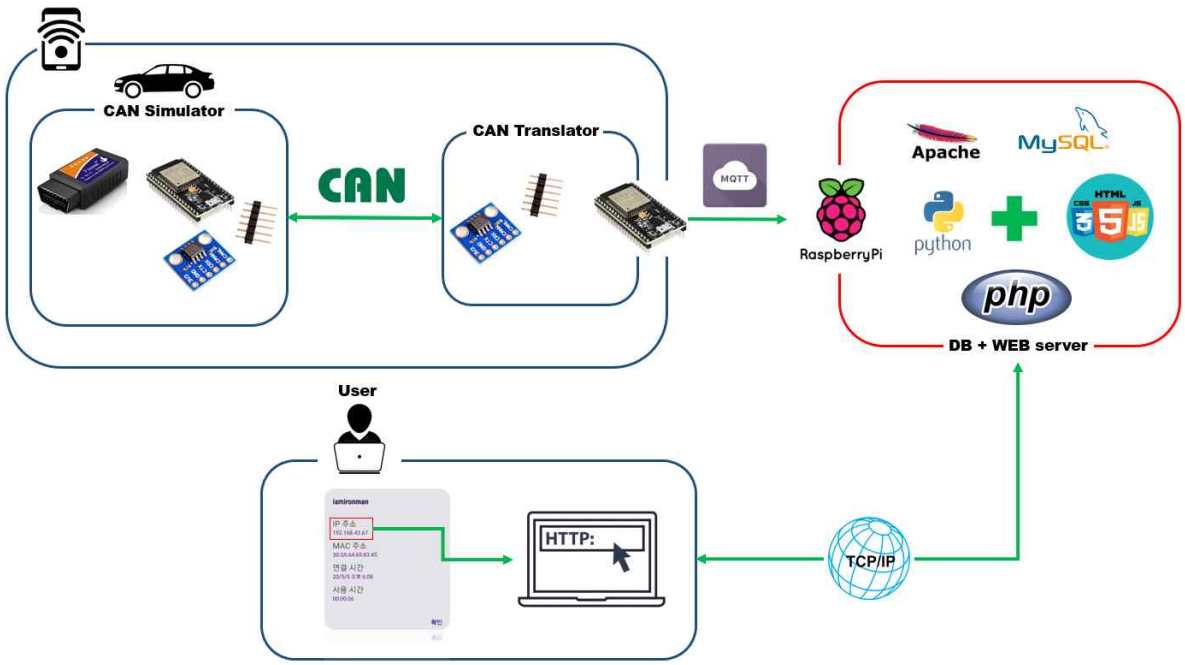
나. 개발배경 및 필요성

ㅇ 차량상태와 진단정보는 차량이 있는 장소에서 진단장비를 차량의 OBD 단자에 접속해야만 얻을 수 있으므로 차량운전자와 정비책임자가 모두 동일시간, 장소 존재해야 함.

ㅇ 접근성에서 가장 뛰어난 수단은 WEB이라 생각했고, WEB 서비스를 통해 차량 정보를 관제할 수 있는 시스템 개발 시작.

ㅇ 이 프로젝트를 통해 차량상태와 진단정보를 MQTT를 이용해서 원격으로 전송함으로서 시간과 공간의 제약을 해결하고 수집된 차량 데이터를 분석하여 부가 서비스를 개발 가능.

다. 작품 구성도



라. 작품의 특징 및 장점

- ㅇ 차량운전자와 정비책임자가 모두 동일시간, 장소에 있어야 하는 공간적, 시간적인 제약에서 벗어나 운전자가 자신의 차량을 점검 / 관리 가능.
- ㅇ 기존 기술인 OBD2를 이용해서 차량상태를 진단 및 데이터 저장.
- ㅇ 수집된 데이터를 웹 페이지에서 시각적으로 표현.
- ㅇ 총 주행거리 데이터를 사용해 차량의 부품 교체 시기 예측.

Ⅱ. 프로젝트 수행결과

가. 업무분장

번호	성명	역할	담당업무
1	최무석	멘 토	- 미팅 계획 수립, 멘토링
3	우창민	팀 장	- SW1, 3D 프린팅
4	정홍석	팀 원2	- SW2
5	강혜인	팀 원3	- HW

나. 주요기능

구분	기능	설명
S/W	UDS 통신	차량상태를 받아오기 위해 차량과 UDS 통신.
	WIFI를 통한 DB와의 연동	받아온 데이터들을 DB에 저장. DB에 저장된 값들을 보여주기 위해 웹서버 활용.
	웹 페이지 상에 출력	실시간 정보, 통계 페이지, 소모품 교체 주기, 고장코드 정보를 웹을 통해 출력
H/W	UDS, HTTP Translator 하드웨어	- ESP32와 CAN transceiver로 구성되어 자동차 CAN 네트워크 데이터를 WiFi 네트워크로 전송
	UDS Simulator 하드웨어	- ESP32와 CAN transceiver로 구성되어 자동차 상태정보, 진단정보를 시뮬레이션

다. 프로젝트 개발환경

구분		항목	적용내역
S/W 개발환경	OS	Window10, Rasbian	개발에 사용된 운영체제
	개발환경(IDE)	Arduino IDE (1.8.9)	esp-32 개발 시 사용
	개발도구	Visual Studio Code	코드 수정 시 사용
	개발언어	C++, Python, PHP, Javascript	웹서버, DB 제작에 사용
H/W 구성장비	디바이스	ESP32, CAN transceiver, 가변저항, 토글 스위치	프로젝트에 사용된 디바이스
	센서	x	x
	통신	CAN, OBD, UDS, MQTT	차량과의 통신을 위해 사용한 통신 프로토콜
	개발언어	C++	esp-32 개발에 사용된 언어

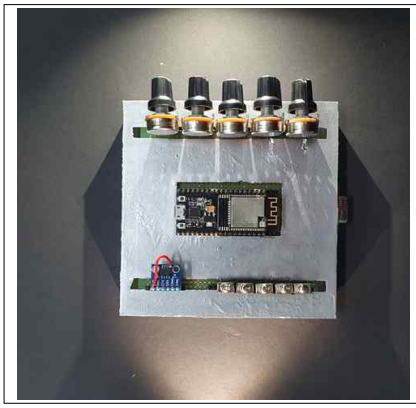
라. 장비(기자재/재료) 활용

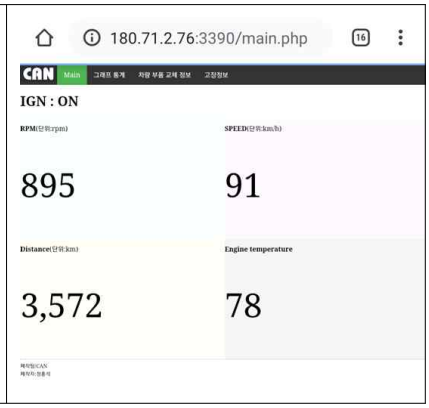
번호	품명	작품에서의 주요기능
1	*ESP-32*	- 시뮬레이터, 트랜스레이터로 사용
2	*CJMCU-230*	- 캔 트랜시버로 사용
3	*라즈베리파이 4*	- 웹서버, DB 구동에 사용
4	*3D 프린터*	-3D 프린팅에 사용

마. 프로그램 작동 동영상

ㅇ https://youtu.be/HxeCA7awFlk

바. 결과물 상세 이미지





사. 달성성과

ㅇ 기존 기획 의도대로 차량상태(차속, 엔진속도, 엔진수온, 밧데리전압), 차량내 ECU의 진단정보(DTC)를 인터넷을 통하여 전송함으로 원격지에서도 차량상태, 진단정보를 확인하고 분석할 수 있는 기반기술을 개발하는 성과를 달성하였음.

■ **논문게재 및 포스터발표**	게재(발표)자명	논문(포스터)명	게재(발표)처	게재(발표)일자
	우창민	IoT를 이용한 차량정보 전송 및 활용 설계 및 구현	한국정보처리학회	2017. 00. 00.
□ **앱(APP) 등록**	등록자명	앱(APP)명	등록처	등록일자
				2017. 00. 00.
□ **프로그램 등록**	등록자명	프로그램명	등록처	등록일자
				2017. 00. 00.
□ **특허/실용신안 출원**	출원자명	특허/실용신안명	출원번호	출원일자
				2017. 00. 00.
□ **기술이전**	기술이전기업명	기술명	금액	이전일자
				2017. 00. 00.
■ **공모전**	구분(교내/대외)	공모전명	수상여부(출품/수상)	상격
	대외	2020 한이음 공모전	입선	
□ **실용화**	*#실용화한 내용에 대한 구체적 작품설명*			
□ **기타**				

Ⅲ. 프로젝트 수행방법

가. 프로젝트 수행일정

구분	추진내용	수행일정								
		3월	4월	5월	6월	7월	8월	9월	10월	11월
계획	Kickoff 미팅, 팀구성, 업무분장, 개발계획		11							
분석	UDS, HTTP gateway 기능분석		20							
설계	HW1,2 회로설계			1						
	SW1 설계			1						
	SW2 설계			1						
개발	HW1,2 개발&3D 프린팅				1					
	SW1 개발					1				
	SW2 개발					1				
	외부 서버 개발									
테스트	HW1과 SW1, HW2와 SW2 통합 테스트							1		
종료	테스트결과 반영, 개선, 결과보고서								31	
오프라인 미팅계획	Skype를 이용한 온라인 미팅		25	9,23	6,20	4,18	2,15, 29	12,2 6	10,2 4	x

나. 문제점 및 해결방안

ㅇ 프로젝트 관리 측면

*문제점

- 하드웨어 가공기술 및 공구부족으로 하드웨어 개발일정 지연.

*해결방안

- 추가 공구(3D 프린터) 구매 및 가공기술 습득을 위해 시간 투자.

ㅇ 작품 개발 측면

*문제점

- CAN통신을 위한 CAN 모듈의 결함으로 인한 프로젝트 개발일정 지연.
- 자동차 정보 수집용 서버를 외부에 설치하는 것으로 변경하여 추가개발일정 발생.

*해결방안

- CAN Tranceiver 모듈의 Vref핀 전압을 증가시킴으로써 문제 해결.
- 라즈베리파이를 이용해 MQTT 서버와 웹서버 구축하여 차량데이터 전송 효율을 높임.

Ⅳ. 기대효과 및 활용분야

ㅇ 차량 데이터를 웹을 통하여 시각적으로 보여줌으로 데이터 접근성 및 활용성 증대.

ㅇ 차량 데이터를 DB를 통하여 체계적으로 관리할 수 있는 기반기술을 제공함으로 차량운행, 정비 관리를 자동화 하여 차량수명 증가 및 관리비용 감소.

ㅇ 차량 운행정보와 차량 정비 데이터를 DB에 저장함으로써 차량배치, 부품 교체 시기 예측과 같은 부가서비스와 연계 가능.

Ⅴ. 참고자료

가. 참고 및 인용자료

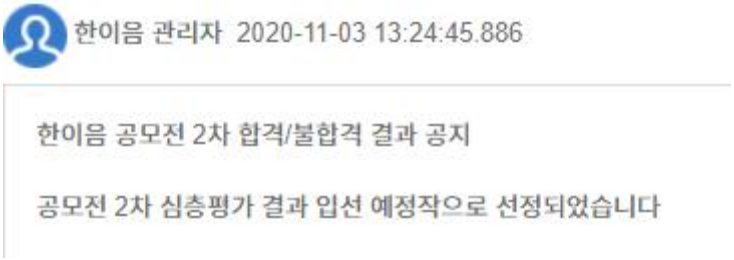
ㅇ Seo-Kyung Lee, Jae-Yong Lee, Dong-Hyun Kim, Kwang-Joo Choi, Jae-IlJung,CAN Communication System using CAN Protocol, Korea Information Processing Society, Republic of Korea, 2006, 4

ㅇ M. Farsi, K. Ratcliff and M. Barbosa, “An overview of controller area network,“ in Computing & Control Engineering Journal, vol. 10, no.3,pp.113-120,June 1999, doi: 10.1049/cce:19990304

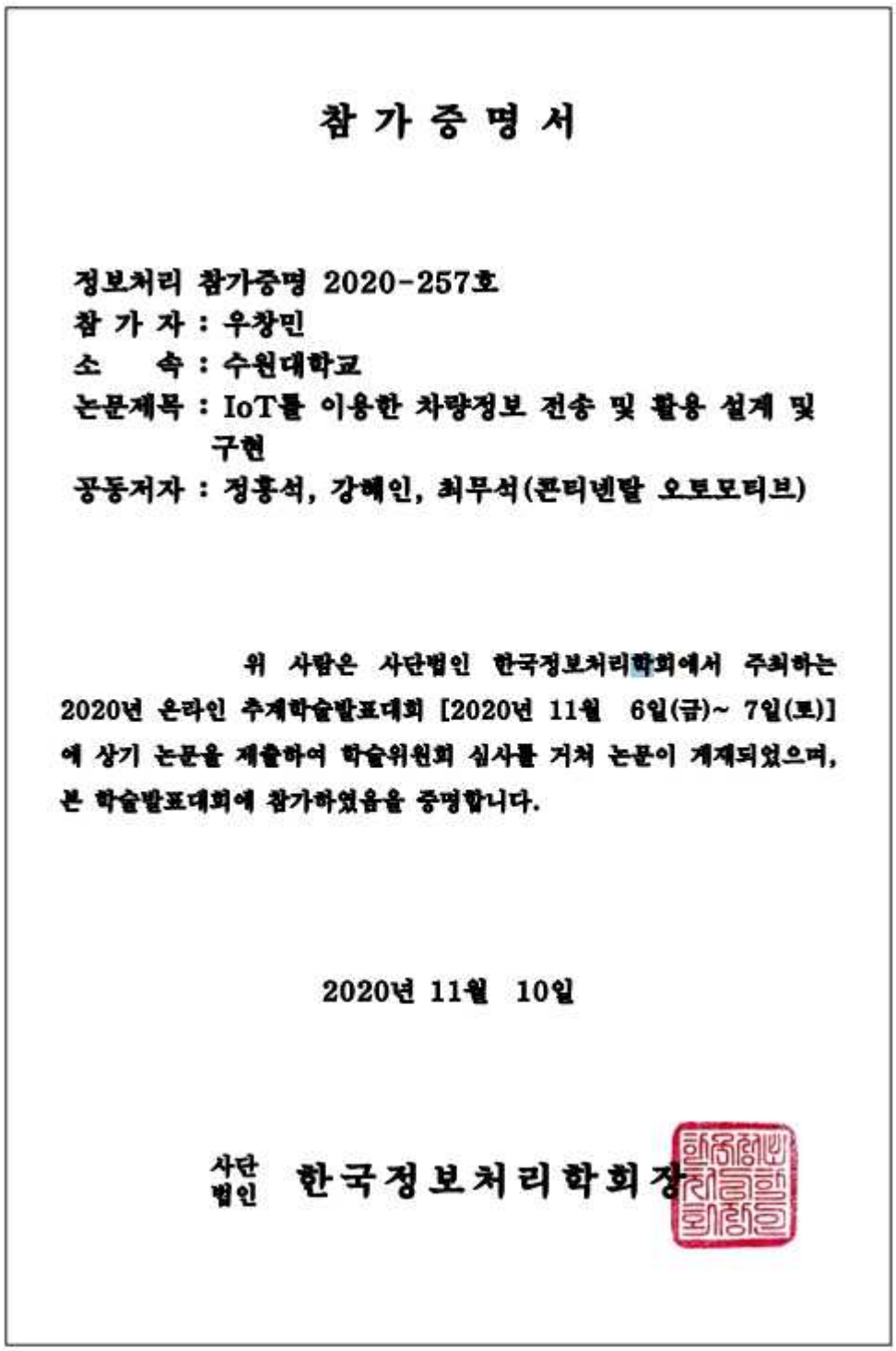
[별첨] 한이음 ICT멘토링 프로젝트 산출물

한이음 ICT멘토링 프로젝트 산출물

1. 한이음 공모전 입선



2. 2020년 온라인 추계학술 발표대회 참가



참 가 증 명 서

정보처리 참가증명 2020-257호
참 가 자 : 우창민
소 속 : 수원대학교
논문제목 : IoT를 이용한 차량정보 전송 및 활용 설계 및 구현
공동저자 : 정홍석, 강혜인, 최무석(콘티넨탈 오토모티브)

위 사람은 사단법인 한국정보처리학회에서 주최하는 2020년 온라인 추계학술발표대회 [2020년 11월 6일(금)~ 7일(토)]에 상기 논문을 제출하여 학술위원회 심사를 거쳐 논문이 게재되었으며, 본 학술발표대회에 참가하였음을 증명합니다.

2020년 11월 10일

사단법인 한국정보처리학회장

3. 소스코드

*CAR Simulator

```
#include <esp32_can.h>
#include <iso-tp-esp32.h>
#include <uds-esp32.h>
ESP32_CAN CAN0;
IsoTp isotp(&CAN0);
UDS uds(&isotp);
struct Session_t session;
//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_pin = 21;
const int Auto_Simulator_pin = 19;
//함수
void AutoSimulator_OBD2(uint32_t Max, uint8_t bitarray[], int digit );
void AutoSimulator_UDS();
void hex_converter_8(uint32_t num, uint8_t bitarray[], int digit);
//PID Data Array
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=100000  =>100000/10=10000km
//DTC Data Array
uint8_t P00BBFuelInjectorInsufficientFlow[] = {0x00, 0xBB};
uint8_t B0020LeftSideAirbagDeploymentControl[] = {0x80, 0x20};
uint8_t C0281BrakeSwitchCircuit[] = {0x42, 0x81};
//PID Struct
#define NUM_PIDS 8
struct {
  uint8_t pid;
  uint8_t* Data;
} responseData[NUM_PIDS] = { {0, pidSupport0x01To0x20},
  {0x05, pid0x05CoolantTemp},
  {0x0C, pid0x0CEngineRpm},
  {0x0D, pid0x0DVehicleSpeed},
  {0x80, pidSupport0x81To0xA0},
  {0x9D, pid0x9DEngineFuelRate},
  {0xA0, pidSupport0xA1To0xC0},
  {0xA6, pid0xA6Odometer}
};
//DTC Struct
#define NUM_DTCS 3
struct {
  int value;
  uint8_t* Data;
} responseDTC[NUM_PIDS] = { {0, P00BBFuelInjectorInsufficientFlow},
  {0, B0020LeftSideAirbagDeploymentControl},
  {0, C0281BrakeSwitchCircuit}
};
void setup() {
  // put your setup code here, to run once:
```

```
  Serial.begin(115200);
  // Initialize ESP32 CAN with a baudrate of 500kb/s and tx pin 16, rx pin 17.
  if (CAN0.begin(16, 17, CAN_500KBPS) == CAN_OK) {
    Serial.println("ESP32 CAN Initialized Successfully!");
  } else {
    Serial.println("Error Initializing ESP32 CAN...");
    return;
  }
  pinMode(5, INPUT); // 고장코드 1
  pinMode(22, INPUT); // 고장코드 2
  pinMode(18, INPUT); // 고장코드 3
  pinMode(21, INPUT); // IGN ON
  pinMode(19, INPUT); //auto simulator
  delay(5000);
}#define LEN_DATA 8
void loop() {
  uint8_t rxData[LEN_DATA];
  uint8_t txData[LEN_DATA];
  struct Session_t diag;
  uint16_t retval = 0;
  uint8_t pid;
  int IGN = digitalRead(IGN_pin);
  if (!IGN) {//IGN OFF
    Serial.println("IGN OFF");
  }
  else {//IGN ON
    Serial.println("IGN ON");
    int AutoSimul = digitalRead(Auto_Simulator_pin);
    if (!AutoSimul) {//Auto Simulator OFF
      Serial.println("Auto Simulator OFF");
      Serial.println("Odometer ");
      int reading_Odometer = analogRead(Odometer_pin);
      Serial.println(reading_Odometer);
      uint32_t Odometer = reading_Odometer * 488; //실질적인 값을 표현하기 위해
      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);
      uint16_t EngineRpm = map(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();
      responseDTC[1].value = digitalRead(Error_Code1_pin);
      responseDTC[2].value = digitalRead(Error_Code2_pin);
      responseDTC[3].value = digitalRead(Error_Code3_pin);
    }
```

```
else {//Auto Simulator ON
  Serial.println("Auto Simulator ON");
  AutoSimulator_OBD2(2000000, pid0xA6Odometer, 8 );
  AutoSimulator_OBD2(255, pid0x05CoolantTemp, 2 );
  AutoSimulator_OBD2(65535, pid0x0CEngineRpm, 4 );
  AutoSimulator_OBD2(255, pid0x0DVehicleSpeed, 2 );
  AutoSimulator_OBD2(60, pid0x9DEngineFuelRate, 2 );
  AutoSimulator_UDS();
}
//Array에 data는 다 들어감
//-> DATA 전송
if (retval = uds.SessionServer(&diag))
{
  Serial.println(F("No OBD request from tool."));
}
else
{
  Serial.print(F("request SID: ")); Serial.println(diag.sid);
  //OBD2 DATA 전송
  if (diag.sid == OBD_MODE_SHOW_CURRENT_DATA) {
    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);
      }
    }
  }
     //UDS 고장진단코드 전송
if (diag.sid == OBD_MODE_READ_DTC) {
  Serial.println("UDS start...");
  int count = 0;// DTC 갯수
  for (int i = 0; i < NUM_DTCS; i++) {
    if (responseDTC[i].value) {
      count++;
    }
  }
  memset(txData, 0, 2 * count);
  if (count == 0) {//DTC가 하나도 없을 때
    Serial.println("No DTC");
    diag.Data = 0;
    diag.len  = 4;
    retval = uds.serverResponse(&diag);
  }
  else { //DTC 1개라도 있을때
    diag.sid = diag.sid + 0x40; // Add 0x40 to received sid
    if (count == 1) {//DTC 1개
      Serial.println(" 1 DTC");
      for (int i = 0; i < NUM_DTCS; i++) {//on인 DTC 1개 찾아서 전송
```

```
            if (responseDTC[i].value) {
              uint8_t lenDTCValue = sizeof(responseDTC[i].Data);
              memcpy(txData, responseDTC[i].Data, lenDTCValue);
              diag.Data = txData;
              diag.len = 2 * count;
              retval = uds.serverResponse(&diag);
            }
          }
        }//DTC 1개 close
        else if (count == 2) {//DTC 2개
          Serial.println(" 2 DTC ");
          for (int i = 0; i < NUM_DTCS; i++) {
            if (responseDTC[i].value) {
              uint8_t lenDTCValue = sizeof(responseDTC[i].Data);
              memcpy(txData + 2 * i, responseDTC[i].Data, lenDTCValue);
            }
          }
          diag.Data = txData;
          diag.len = 4;
          retval = uds.serverResponse(&diag);
        }//DTC 2개 close
        else if (count == 3) {//DTC 3개
          Serial.println(" 3 DTC ");
          for (int i = 0; i < NUM_DTCS; i++) {
            uint8_t lenDTCValue = sizeof(responseDTC[i].Data);
            memcpy(txData + 2 * i, responseDTC[i].Data, lenDTCValue);
          }
          diag.Data = txData;
          diag.len = 6;
          retval = uds.serverResponse(&diag);
        }//DTC 3개 close
      }//DTC 1개라도 있을때 close
    }//sid=dtc close
    }
    delay(1000);
  }
}void AutoSimulator_OBD2(uint32_t Max, uint8_t bitarray[], int digit ) {
  uint32_t value = random(0, Max);
  hex_converter_8(value, bitarray, digit);
}void AutoSimulator_UDS() {
  for (int i = 0; i < NUM_DTCS; i++) {
    responseDTC[i].value = random(0, 2);
  }
}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;
  Serial.println();
  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(temp[pos - i - 1]); Serial.print(" ");
  }
  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
    if (bitarray[j] < 16) {
      Serial.print("0");
    }
    Serial.print(bitarray[j], HEX); Serial.print(" ");
  }
```

```
}
```

*CAR Translator

```
# include <esp32_can.h>
# include <iso-tp-esp32.h>
# include <uds-esp32.h>
# include <WiFi.h>
# include <PubSubClient.h>
# define MSG_BUFFER_SIZE (50)
# define BUILTIN_LED 25
ESP32_CAN CAN0;
IsoTp isotp(&CAN0);
UDS uds(&isotp);
struct Session_t session;
char msg[MSG_BUFFER_SIZE];
const char* ssid = "Hong";
const char* password = "123456789q";
const char* mqtt_server = "180.71.2.76";
uint8_t tmp[] = {};
String packet;
long lastMsg = 0;
// 웹에 나타나는 정보 변수 초기화
int IGN = 0;
int RPM = 0;
int D = 0;
int V = 0;
int Eng_temp = 0;
int Fuel_level = 0;
WiFiClient esp_client;
PubSubClient client(esp_client);
// 사용되는 함수 초기화
void displayWifistatus(void);
void OBD2_receiver(uint8_t pid);
void UDS_receiver(void);
long data_process(int count, uint8_t *array_pointer);
void mqtt_trans(int IGN, int (RPM), double D, int V, float Eng_temp, int Fuel_level);
void callback(char* topic, byte* payload, unsigned int length);
void reconnect();
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println();
  //1. wifi연결
  WiFi.begin(ssid, password);
  Serial.println("Connecting");
  while (WiFi.status() != WL_CONNECTED)
  {
    displayWifistatus();
    delay(500);
  }
  displayWifistatus();
  Serial.println();
  Serial.print("Conncected, IP address: ");
  Serial.println(WiFi.localIP());
  Serial.print("Gateway IP: ");
  Serial.println(WiFi.gatewayIP());
  Serial.print("RSSI: ");
  Serial.println(WiFi.RSSI());
  //2. Mqtt 연결
  pinMode(BUILTIN_LED, OUTPUT);
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

  //3. CAN 연결
```

```
  if (CAN0.begin(16, 17, CAN_500KBPS) == CAN_OK) {          //25,50,100,125,250,500,800,1000
    Serial.println("ESP32 CAN Initialized Successfully!");
  } else {
    Serial.println("Error Initializing ESP32 CAN...");
    return;
  }
  delay(5000);
}void loop() {
  uint8_t pid[] = {0x1F, 0x05, 0x0C, 0x0D, 0x9D, 0xA6};
  uint8_t uds_id[] = {};
  //1. OBD2로 Data receive
  for (int i = 0; i < sizeof(pid) / sizeof(uint8_t) ; i++)
    OBD2_receiver(pid[i]);
  //2. UDS로 Data receive
  UDS_receiver();
  if (RPM != 0)    //엔진이 켜져있다면
    IGN = 1;        //IGN을 True로
  else              //엔진이 꺼져있다면
    IGN = 0;        //IGN을 False로
  if (IGN == 1);     //3. 시동이 켜져있다면
    mqtt_trans(IGN);    //4. MQTT를 이용해 서버로 전송
}void displayWifistatus(void) {
  Serial.print("ESP WiFi connection status: ");
  switch (WiFi.status()) {
    case 0:
      Serial.println("WL_IDLE_STATUS");
      break;
    case 1:
      Serial.println("WL_NO_SSID_AVAIL");
      break;
    case 3:
      Serial.println("WL_CONNECTED");
      break;
    case 4:
      Serial.println("WL_CONNECT_FAILED");
      break;
    case 6:
      Serial.println("WL_DISCONNECTED");
      break;
    default:
      Serial.println("Unknown status");
  }
}void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
  // Switch on the LED if an 1 was received as first character
  if ((char)payload[0] == '1') {
    digitalWrite(BUILTIN_LED, LOW);    // Turn the LED on (Note that LOW is the voltage level
    // but actually the LED is on; this is because
    // it is active low on the ESP-01)
  } else {
    digitalWrite(BUILTIN_LED, HIGH);  // Turn the LED off by making the voltage HIGH
  }
}void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
```

```
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish("start", "Connected Mqtt...");
      // ... and resubscribe
      client.subscribe("inTopic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}void OBD2_receiver(uint8_t pid) {
  int num = 0;
  int count = 0;
  long f_result = 0;
  struct Session_t diag;
  uint16_t retval = 0;
  Serial.print(F("OBD2 request PIDs: ")); Serial.println(pid);
  for (int i = 0; i < 1; i++) {
    uint16_t rx_id_array[] = {0x7E9, 0x7E9};
    diag.tx_id = 0x7DF; //0x7E0, 0x7DF  -> 7DF였으면 원래 응답이 와야하는대 안됐음.
    //                    예제 하나 줄테니 실제 차량에서 진행해보자.
    //
    diag.rx_id = rx_id_array[i];
    Serial.print("tx_id : "); Serial.println(diag.tx_id);
    Serial.print("rx_id : "); Serial.println(diag.rx_id);
    //diag.rx_id = 0x7E8;
    diag.sid = OBD_MODE_SHOW_CURRENT_DATA;
    diag.Data = &pid;
    diag.len = 1;
    if (retval = uds.Session(&diag))
    {
      Serial.print(F("OBD2 Error "));
      Serial.print(retval);
      Serial.print(F(" NRC "));
      Serial.println(retval, HEX);
    }
    else
    {
      //0x1F, 0x05, 0x0C, 0x0D, 0x9D, 0xA6
      //pid별 최대 바이트로 자르는 코드 시작
      //1.각 pid별로 몇바이트인지 count변수에 저장
      switch (pid) {
        case 0x05:
          count = 1;
          //2. count만큼 데이터를 자르고 저장 tmp배열에 임시 저장
          for (int i = 0; i < count; i++) {
            tmp[i] = diag.Data[i];
          }
          //3. 데이터 후처리
          Eng_temp = data_process(count, tmp) - 40;
          Serial.println("엔진 온도");
          Serial.println(Eng_temp);
          break;
        case 0x0C:
          count = 2;
          //2. count만큼 데이터를 자르고 저장 tmp배열에 임시 저장
          for (int i = 0; i < count; i++) {
            tmp[i] = diag.Data[i];
```

```
            }
            //3. 데이터 후처리
            RPM = data_process(count, tmp) / 4;
            Serial.println("RPM");
            Serial.println(RPM);
            break;
          case 0x0D:
            count = 1;
            //2. count만큼 데이터를 자르고 저장 tmp배열에 임시 저장
            for (int i = 0; i < count; i++) {
              tmp[i] = diag.Data[i];
            }
            //3. 데이터 후처리
            V = data_process(count, tmp);
            Serial.println("차량 속력");
            Serial.println(V);
            break;
          case 0x9D:
            count = 1;
            //2. count만큼 데이터를 자르고 저장 tmp배열에 임시 저장
            for (int i = 0; i < count; i++) {
              tmp[i] = diag.Data[i];
            }
            //3. 데이터 후처리
            Fuel_level = data_process(count, tmp);
            Serial.println("연료 잔량");
            Serial.println(Fuel_level);
            break;
          case 0xA6:
            count = 4;
            //2. count만큼 데이터를 자르고 저장 tmp배열에 임시 저장
            for (int i = 0; i < count; i++) {
              tmp[i] = diag.Data[i];
            }
            //3. 데이터 후처리
            D = data_process(count, tmp) / 10;
            Serial.println("총 주행거리");
            Serial.println(D);
            break;
          default:
            Serial.println("미구현");
            break;
        }
      }
      Serial.print("OBD code : "); Serial.print(pid); Serial.println(" end");
      Serial.println("");
      delay(1000);
    }
  }void UDS_receiver(void) {
    int num = 0;
    uint8_t pid = 0x00;  //pid 의미없음
    struct Session_t diag;
    uint16_t retval = 0;
    Serial.println(F("Diag Session"));
    for (int i = 0; i < 8; i++) {
      uint16_t rx_id_array[] = {0x7E8, 0x7E9, 0x7EA, 0x7EB, 0x7EC, 0x7ED, 0x7EE, 0x7EF};
      diag.tx_id = 0x7DF; //0x7E0, 0x7DF
      diag.rx_id = rx_id_array[i];
      Serial.print("tx_id : "); Serial.println(diag.tx_id);
      Serial.print("rx_id : "); Serial.println(diag.rx_id);
      //diag.tx_id = 0x7E0;
      //diag.rx_id = 0x7E8;
      diag.sid = OBD_MODE_READ_DTC;
      diag.Data = &pid;            //진단 데이터 수신
```

```
    diag.len = 0;                 // before 1
    if (retval = uds.Session(&diag))
    {
      Serial.print(F("UDS Session Error "));
      Serial.print(retval); Serial.print(F(" NRC "));
      Serial.println(retval, HEX);
    }
    else
    {
      Serial.println(F("Established with "));
      uds.print_buffer(diag.Data, diag.len);
      Serial.print("Length : "); Serial.println(diag.len);
    }
    Serial.print("UDS code"); Serial.print(num); Serial.println(" end...");
    delay(1000);
  }
}long data_process(int count, uint8_t *array_pointer) {
  long result = 0;
  //3. 데이터 후처리 시작
  if (count != 1) //0,2,4,8씩 제곱
  {
    if (count == 2) //0~2
    {
      result = tmp[0] * pow(16, 2) + tmp[1];
      //Serial.print("count is 2 : "); Serial.println(result);
    }
    else if (count == 3) //0~4
    {
      result = tmp[0] * pow(16, 4) + tmp[1] * pow(16, 2) + tmp[2];
      //Serial.print("count is 3 : "); Serial.println(result);
    }
    else if (count == 4) //0~8
    {
      result = tmp[0] * pow(16, 8) + tmp[1] * pow(16, 4) + tmp[2] * pow(16, 2) + tmp[3];
      //Serial.print("count is 4 : "); Serial.println(result);
    }
  }
  else {
    result = tmp[0];
    //Serial.print("count is 1 : "); Serial.println(tmp[0]);
  }
  return result;
}void mqtt_trans(int IGN, int RPM, int D, int V, int Eng_temp, int Fuel_level) {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  unsigned long now = millis();
  if (now - lastMsg > 500) {
    lastMsg = now;
    packet = String(IGN)+" "+String(RPM)+" "+String(D)+" "+String(V)+" "+String(Eng_temp)+" "+String(Fuel_level);
    packet.toCharArray(msg, 50);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish("Env_car", msg);
  }
}
```

*CAR Translator_실제 차 통신에 쓰이는 코드

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

```
# define MSG_BUFFER_SIZE (50)
# define BUILTIN_LED 25
ESP32_CAN CAN0;
IsoTp isotp(&CAN0);
UDS uds(&isotp);
struct Session_t session;
char msg[MSG_BUFFER_SIZE];
const char* ssid = "Hong";
const char* password = "123456789q";
const char* mqtt_server = "180.71.2.76";
uint8_t tmp[] = {};
String packet;
long lastMsg = 0;
// 웹에 나타나는 정보 변수 초기화
int IGN = 0;
int RPM = 0;
int D = 0;
int V = 0;
int Eng_temp = 0;
int Fuel_level = 0;
int Eng_runtime = 0;
int Fuel_tank_level_input = 0;
int engine_load = 0;
int DTC_on_D = 0;
int DTC_clr_D = 0;
int engine_oil = 0;
int relative_accel_pedle = 0;
WiFiClient esp_client;
PubSubClient client(esp_client);
// 사용되는 함수 초기화
void displayWifistatus(void);
void OBD2_receiver(uint8_t pid);
void UDS_receiver(void);
long data_process(int count, uint8_t *array_pointer);
void mqtt_trans(int IGN, int (RPM), double D, int V, float Eng_temp, int Fuel_level);
void callback(char* topic, byte* payload, unsigned int length);
void reconnect();
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println();
  //1. wifi연결
  WiFi.begin(ssid, password);
  Serial.println("Connecting");
  while (WiFi.status() != WL_CONNECTED)
  {
    displayWifistatus();
    delay(500);
  }
  displayWifistatus();
  Serial.println();
  Serial.print("Conncected, IP address: ");
  Serial.println(WiFi.localIP());
  Serial.print("Gateway IP: ");
  Serial.println(WiFi.gatewayIP());
  Serial.print("RSSI: ");
  Serial.println(WiFi.RSSI());
  //2. Mqtt 연결
  pinMode(BUILTIN_LED, OUTPUT);
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  //3. CAN 연결
  if (CAN0.begin(16, 17, CAN_500KBPS) == CAN_OK) {          //25,50,100,125,250,500,800,1000
    Serial.println("ESP32 CAN Initialized Successfully!");
```

```
  } else {
    Serial.println("Error Initializing ESP32 CAN...");
    return;
  }
  delay(5000);
}void loop() {
  uint8_t pid[] = {0x04, 0x05, 0x0C, 0x0D, 0x11, 0x1F, 0x21, 0x2F, 0x31, 0x5A, 0x5C};
  uint8_t uds_id[] = {};
  //1. OBD2로 Data receive
  for (int i = 0; i < sizeof(pid) / sizeof(uint8_t) ; i++)
    OBD2_receiver(pid[i]);
  //2. UDS로 Data receive
  UDS_receiver();
  //3. 시동이 켜져있다면
  if (IGN == 1);
    mqtt_trans(IGN, RPM, D, V, Eng_temp, Fuel_level);    //4. MQTT를 이용해 서버로 전송
}void displayWifistatus(void) {
  Serial.print("ESP WiFi connection status: ");
  switch (WiFi.status()) {
    case 0:
      Serial.println("WL_IDLE_STATUS");
      break;
    case 1:
      Serial.println("WL_NO_SSID_AVAIL");
      break;
    case 3:
      Serial.println("WL_CONNECTED");
      break;
    case 4:
      Serial.println("WL_CONNECT_FAILED");
      break;
    case 6:
      Serial.println("WL_DISCONNECTED");
      break;
    default:
      Serial.println("Unknown status");
  }
}void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
  // Switch on the LED if an 1 was received as first character
  if ((char)payload[0] == '1') {
    digitalWrite(BUILTIN_LED, LOW);    // Turn the LED on (Note that LOW is the voltage level
    // but actually the LED is on; this is because
    // it is active low on the ESP-01)
  } else {
    digitalWrite(BUILTIN_LED, HIGH);  // Turn the LED off by making the voltage HIGH
  }
}void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      // Once connected, publish an announcement...
```

```
      client.publish("start", "Connected Mqtt...");
      // ... and resubscribe
      client.subscribe("inTopic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}void OBD2_receiver(uint8_t pid) {
  int num = 0;
  int count = 0;
  long f_result = 0;
  struct Session_t diag;
  uint16_t retval = 0;
  Serial.print(F("OBD2 request PIDs: ")); Serial.println(pid);
  for (int i = 0; i < 1; i++) {
    uint16_t rx_id_array[] = {0x7E9, 0x7E9};
    diag.tx_id = 0x7DF; //0x7E0, 0x7DF  -> 7DF였으면 원래 응답이 와야하는대 안됐음.
    //                  예제 하나 줄테니 실제 차량에서 진행해보자.
    //
    diag.rx_id = rx_id_array[i];
    Serial.print("tx_id : "); Serial.println(diag.tx_id);
    Serial.print("rx_id : "); Serial.println(diag.rx_id);
    //diag.rx_id = 0x7E8;
    diag.sid = OBD_MODE_SHOW_CURRENT_DATA;
    diag.Data = &pid;
    diag.len = 1;
    if (retval = uds.Session(&diag))
    {
      Serial.print(F("OBD2 Error "));
      Serial.print(retval);
      Serial.print(F(" NRC "));
      Serial.println(retval, HEX);
    }
    else
    {
      //0x1F, 0x05, 0x0C, 0x0D, 0x9D, 0xA6
      //pid별 최대 바이트로 자르는 코드 시작
      //diag.Data는 한 프레임에 8비트씩 담겨진다.
      /*
          Serial.println(diag.Data[0]);///A 8bit
          Serial.println(diag.Data[1]);///B 8bit
          Serial.println(diag.Data[2]);///C 8bit
          Serial.println(diag.Data[3]);///D 8bit
      */
      //1.각 pid별로 몇바이트인지 count변수에 저장
      switch (pid) {
        case 0x04:
          engine_load = 100*diag.Data[0]/255;
          Serial.print("엔진 부하 : ");
          Serial.println(engine_load);
          break;
        case 0x05:
          Eng_temp = diag.Data[0]-40;
          Serial.print("엔진 온도 : ");
          Serial.println(Eng_temp);
          break;
        case 0x0C:
          RPM = ((diag.Data[0]*pow(2,8)) + diag.Data[1])/4;
          Serial.print("RPM : ");
          Serial.println(RPM);
```

```
            break;
          case 0x0D:
            V = diag.Data[0];
            Serial.print("차량 속력 : ");
            Serial.println(V);
            break;
          case 0x1F: //IGN 체크
            Eng_runtime = (diag.Data[0]*pow(2,8)) + diag.Data[1];
            Serial.print("엔진이 켜진 이후부터의 주행 시간 : ");
            Serial.println(Eng_runtime);
            if(Eng_runtime > 0)
              IGN = 1;
            else
              IGN = 0;
            break;
          case 0x21:
            DTC_on_D = (diag.Data[0]*pow(2,8)) + diag.Data[1];
            Serial.print("DTC_on_D : ");
            Serial.println(DTC_on_D);
            break;
          case 0x2F:
            Fuel_tank_level_input = 100*diag.Data[0]/255;
            Serial.print("Fuel_tank_level_input : ");
            Serial.println(Fuel_tank_level_input);
            break;
          case 0x31:
            DTC_clr_D = (diag.Data[0]*pow(2,8)) + diag.Data[1];
            Serial.print("DTC_clr_D : ");
            Serial.println(DTC_clr_D);
            break;
          case 0x5A:
            relative_accel_pedle = 100*diag.Data[0]/255;
            Serial.print("relative_accel_pedle : ");
            Serial.println(relative_accel_pedle);
            break;
          case 0x5C:
            engine_oil = diag.Data[0]-40;
            Serial.print("engine_oil : ");
            Serial.println(engine_oil);
            break;
          case 0x9D:
            Fuel_level = 0;
            Serial.print("연료 잔량 : ");
            Serial.println(Fuel_level);
            break;
          case 0xA6:
            D = (diag.Data[0]*pow(2,24)+diag.Data[1]*pow(2,16)+diag.Data[2]*pow(2,8)+diag.Data[3])/10;
            Serial.print("총 주행거리");
            Serial.println(D);
            break;
          default:
            Serial.println("미구현");
            break;
        }
      }
      Serial.print("OBD code : "); Serial.print(pid); Serial.println(" end");
      Serial.println("");
      delay(1000);
    }
}void UDS_receiver(void) {
    int num = 0;
    uint8_t pid = 0x00;   //pid 의미없음
    struct Session_t diag;
    uint16_t retval = 0;
```

```
  Serial.println(F("Diag Session"));
  for (int i = 0; i < 8; i++) {
    uint16_t rx_id_array[] = {0x7E8, 0x7E9, 0x7EA, 0x7EB, 0x7EC, 0x7ED, 0x7EE, 0x7EF};
    diag.tx_id = 0x7DF; //0x7E0, 0x7DF
    diag.rx_id = rx_id_array[i];
    Serial.print("tx_id : "); Serial.println(diag.tx_id);
    Serial.print("rx_id : "); Serial.println(diag.rx_id);
    //diag.tx_id = 0x7E0;
    //diag.rx_id = 0x7E8;
    diag.sid = OBD_MODE_READ_DTC;
    diag.Data = &pid;            //진단 데이터 수신
    diag.len = 0;                // before 1
    if (retval = uds.Session(&diag))
    {
      Serial.print(F("UDS Session Error "));
      Serial.print(retval); Serial.print(F(" NRC "));
      Serial.println(retval, HEX);
    }
    else
    {
      Serial.println(F("Established with "));
      uds.print_buffer(diag.Data, diag.len);
      Serial.print("Length : "); Serial.println(diag.len);
    }
    Serial.print("UDS code"); Serial.print(num); Serial.println(" end...");
    delay(1000);
  }
}void mqtt_trans(int IGN, int RPM, int D, int V, int Eng_temp, int Fuel_level) {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  unsigned long now = millis();
  if (now - lastMsg > 500) {
    lastMsg = now;
    packet = String(IGN)+" "+String(RPM)+" "+String(D)+" "+String(V)+" "+String(Eng_temp)+" "+String(Fuel_level);
    packet.toCharArray(msg, 50);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish("Env_car", msg);
  }
}
```