TI DSP, MCU, Xilinx FPGA 기반의 자율주행 자동차

이름 : 안상재
E-mail : sangjae2015@naver.com
연락처 : 010-4147-2573

- 프로필 -

안 상 재 (AHN SANG JAE)

- 대전 유성고등학교 졸업 (2010. 02)
- 한국항공대학교 항공전자정보공학부 졸업예정 (2019. 02)

- 2018년 02월 ~ 2018년 10월 사물인터넷 기반의 임베디드 개발자 과정 수료
(한국아이티 인재개발원, 국비지원 교육)

-> 본 과정의 이론수업에 대한 github 링크 :
https://github.com/SHL-Education/Homework/tree/master/sangjaeahn
-> 본 과정의 프로젝트 진행 중 만든 문서에 관한 github 링크 :
https://github.com/SHL-Third-Education/Project#sangjae-ahn

- 가치관 : 개발자라는 직업을 단순히 돈벌이 수단이 아닌 평생 저의 업으로 생각하고,
큰 자부심을 느낍니다. 항상 최고가 되기 위해 꾸준한 자기계발과 매사에 성실이 임하겠습니다.

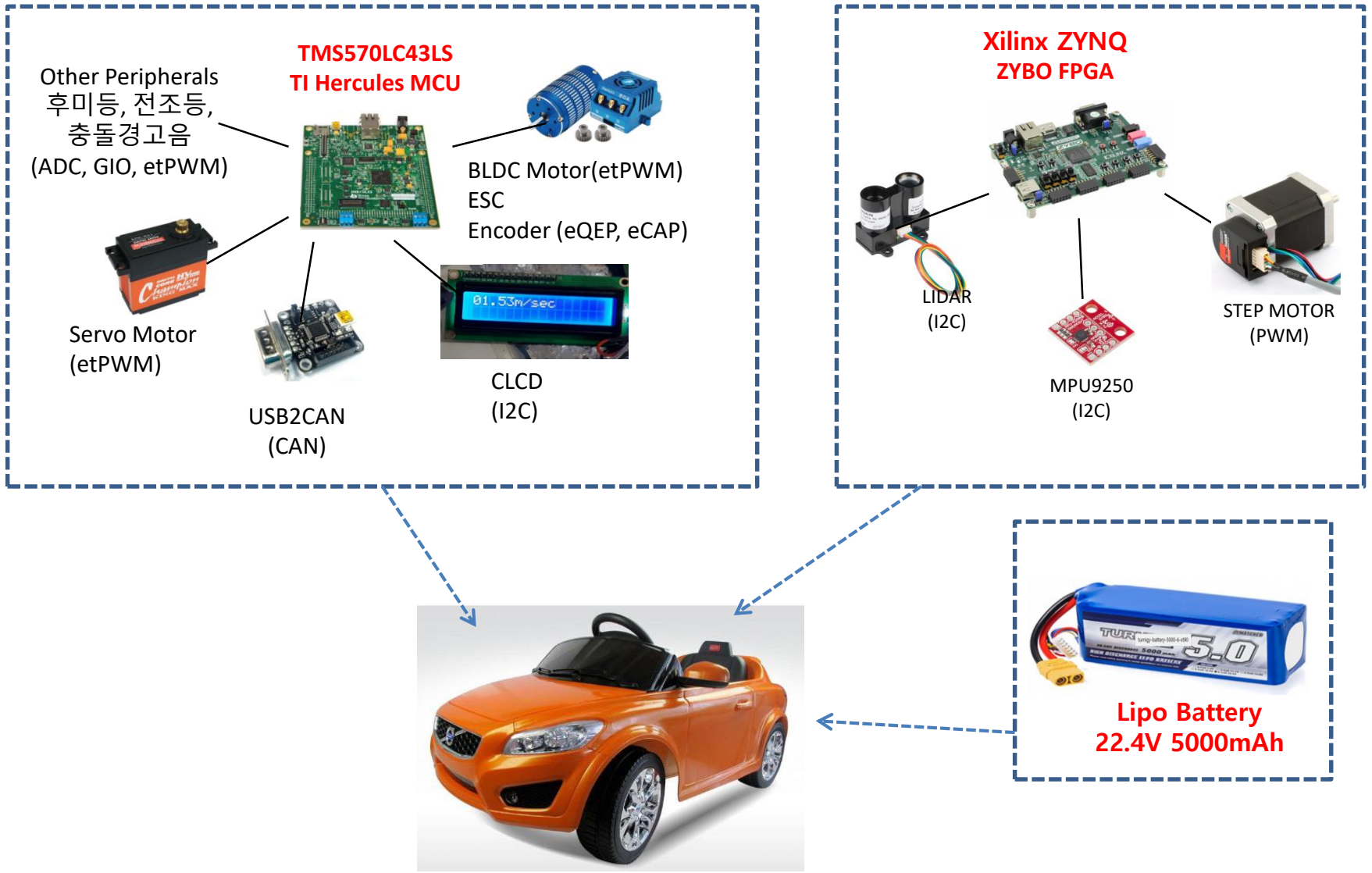
- 보유 기술 : 아날로그/디지털 회로설계(중), TI, ST, AVR MCU 펌웨어 프로그래밍(중상)
PCB 아트웍 하(Altium)

- 목 차 -

1. 전체 프로젝트 구성

1-1. 프로젝트 진행 중 TEST 모습

1-2. Hardware Architecture



1-3. Development Environments

MCU : TI Hercules MCU (TMS570LC43xx)
Compiler : CCS v8.1.0, HalCoGen v04.07.00
Language : C



FPGA : Xilinx Zynq Zybo
PL : Vivado 2017.4
PS : petalinux v2015.4, Vivado SDK
Language : Verilog, C




협업 Tool : GitHub, Slack


1-4. BOM LIST

Bill of Materials

*. 마지막 수정일: 2018-10-03

	구매물품	개수	환율	구매가격	합계	링크
BLDC모터 + ESC	ESC Program Card	1		₩ 19,565	₩ 19,565	https://hobbyking.com/en_us/hobbykingr-tm-x-car-beast-series-lcd-program-card.html
	XT90 Connectors Male With End Cap	1		₩ 11,959	₩ 11,959	https://hobbyking.com/en_us/nylonxt90connectorsmale-5pcs-bag.html
	Gold Connectors 10 pairs (20pc)	1		₩ 3,391	₩ 3,391	
	10T/5mm Steel Pinion Gear	1		₩ 2,509	₩ 2,509	https://hobbyking.com/en_us/10t-5mm-m1-hardened-steel-pinion-gear-1pc.html
	XERUN 4274SD 2200KV Sensored Motor + 150A ESC	1	1146	₩ 200,319	₩ 200,319	https://amainhobbies.com/
	Hobbywing 11T 5mm M1 STEEL PINION GEAR	1		₩ 11,500	₩ 11,500	https://amainhobbies.com/
	ESC 납땜 공임비	1		₩ 10,000	₩ 10,000	http://buyrc.co.kr/
배터리	(엘란AI)배터리 충전기 + 커넥터	1	1130	₩ 46,750	₩ 46,750	https://hobbyking.com/en_us/fire-retardant-lipo-battery-bag-220x155x115mm-with-handle.html
	TURNIGY® FIRE RETARDANT LIPOLY BATTERY BAG (ZIPPERED) (200X155X95MM) (1PC)	1		₩ 9,687	₩ 9,687	https://hobbyking.com/en_us/turnigy-5000mah-6s-40c-lipo-pack-xt90.html
	Turnigy 5000mAh 6S 40C Lipo Pack w/ XT90	1		₩ 68,315	₩ 68,315	
	HobbyKing™ Lipo Voltage Checker (2S~8S)	1		₩ 2,385	₩ 2,385	https://hobbyking.com/en_us/hobbykingtm-lipo-voltage-checker-2s-8s.html
서보모터	Gear	1		₩ 10,000	₩ 10,000	
	KINGMAX DCS16942CHV	1		₩ 53,760	₩ 53,760	
전원부	전력분배보드	1		₩ 8,400	₩ 8,400	
	12V 5A 벽컨버터(96W)	1		₩ 20,091	₩ 20,091	
	DCDC Converter	6		₩ 6,300	₩ 6,300	
스텝모터	Step Motor Driver	1		₩ 33,000	₩ 33,000	
	Step Motor	1		₩ 121,000	₩ 121,000	
Module	CDS	3		₩ 270	₩ 270	http://www.ic114.com/WebSite/site/sc/00V0.aspx?id_p=P0047495
	MPU6050	1		₩ 1,750	₩ 1,750	
	MPU9250	1		₩ 11,000	₩ 11,000	http://mechasolution.com/shop/goods/goods_view.php?goodsno=330567&inflow=naver&NaPm=ct%3Djef08k%7Cci%3Db5850244e0a855713d2fb80d9c7258c4b4a1c5b9%7Ctr%3Dslsl%7Csn%3D188145%7Chk%3Dd1a00e82317c752ec552ccef5cfe0706d878ac483
	USB2CAN	1		₩ 80,080	₩ 80,080	
	PMOD CAN	1		₩ 22,000	₩ 22,000	
	Lidar	1		₩ 174,000	₩ 174,000	
기타	Micro SDCard(16G) + 리더기	1		₩ 18,050		
	슬립링 1000rpm	1		₩ 12,360		
	LED 콘리	2		₩ 24,400		http://smartstore.naver.com/domekit/products/2918518787?NaPm=ct%3Djg4y4u0%7Cci%3Deb958cdbdab0dcfffdb5302d154091c3e28909c3%7Ctr%3Dsls%7Csn%3D278114%7Chk%3D71a22f3c1f566425630973853da1faeecc005647
	아크릴 주문제작	1		₩ 60,000		
	차량 프레임	1		₩ 50,000		
	용수철	1		₩ 5,000		
	Micro SDCard(32G)	1		₩ 20,000		
	Power LED	6		₩ 9,320		https://smartstore.naver.com/thecamping/products/754703737?NaPm=ct%3Djkt3ud15%7Cci%3Dcheckout%7Ctr%3Dsls|_myc%7Ctrx%3D%7Chk%3D39b36343a5eecacfb380694e077b14462ca898a0
	라이다 아크릴+기어	1		₩ 22,000		
	주행코스용 검은노끈	5		₩ 9,490	₩ 12,490	https://smartstore.naver.com/thecamping/products/754703737?NaPm=ct%3Djkt3ud15%7Cci%3Dcheckout%7Ctr%3Dsls|_myc%7Ctrx%3D%7Chk%3D39b36343a5eecacfb380694e077b14462ca898a0
	9V 건전지	6		₩ 8,000	₩ 8,000	
	MCU, FPGA용 배럴잭	1		₩ 3,000	₩ 38,000	
	십각대	1		₩ 5,000	₩ 5,000	
Board	TI Hercules MCU			₩ 221,089	₩ 221,089	
	Xilinx Zybo FPGA			₩ 258,500	₩ 258,500	
총 구매금액					₩ 1,471,110	

2. USB to CAN 모듈을 활용한 통신제어

*** 사용 모듈**
MoonWalker MW USB2CAN(FIFO) v2

*** 구현방법**
CAN은 I2C, SPI, UART 통신에 비해 노이즈에 강한 특성이 있다.
인터럽트방식을 사용하여 CAN 통신을 구현하였다.
다음과 같은 통신 프로토콜을 지정하여 사용하였다.
CAN으로 보내는 데이터는
명령어 2비트 후속데이터 4비트로 구성하며,
명령어에 따라 후속데이터는 다른 형태를 가진다.

Signal	Command	Data
좌회전	1	-
우회전	2	-
정지	5	-
충돌경보	6	-
좌측깜빡이	7	0 or 1
우측깜빡이	8	0 or 1
전조등	9	0 or 1
BLDC 속도 지정	12	PWM Duty
조향 각도지정	13	PWM Duty

MCU와 DSP 간의 커스텀 CAN 프로토콜

*** Issue사항**
ESP8266을 제외시켰다.
처음에는 안드로이드폰과 MCU간에 Wifi Module인 ESP8266을 사용하였다. ESP8266의 펌웨어를 업그레이드 하고 UART를 이용하여 AT Command를 보내서 제어하였지만, 송수신 값이 불안정하여 사용하지 않았다.

두번째 방법으로 DSP에 Wifi Antenna를 추가하였다.
TCP/IP서버를 오픈하여 클라이언트의 핸드폰에서 데이터를 보내면 DSP는 받은 데이터를 CAN을 통해 MCU로 전송한다.

2-1. 고 찰

HALCoGen 툴에서 제공하는 CAN 통신의 예제 코드를 참조하면서 MCU의 CAN 통신을 구현했다. 처음에는 DSP 보드의 CAN 통신 구현이 안되어 있는 상태이었기 때문에, 대신 PC 에서 MCU로 데이터를 전송하면서 TEST를 하였다.

CAN모듈과 MCU 사이를 점퍼선으로 연결해서 TEST를 했지만, 통신이 잘 되지 않았다. 통신이 아예 되지 않는 것도 아니고, 됐다 안됐다가를 반복하는 현상이 발생했다. 처음에는 통신이 잘 안되는 원인을 정확히 알지 못하고, 소스 코드 상의 문제라고만 판단했다. 소스 코드만 오랜 시간 들여다 보면서 문제를 해결하려고 했지만, 여전히 통신은 잘 되지 않았고, 우연히 MCU 보드의 CAN 통신 단자와 점퍼선이 연결이 느슨하게 되었다는 사실을 발견했다. 그리고 MCU 보드의 CAN 통신 단자에 점퍼선을 조일 수 있는 단자가 있다는 것을 보고, 일자 드라이버로 점퍼선을 잘 조이니 회로 연결이 안정적으로 되었고, 그 결과 통신이 잘 안되던 문제를 해결할 수 있었다.

내가 만든 충돌 경보 사이렌 회로, BLDC 모터 제어 뿐만이 아니라 다른 팀원들이 구현한 기능들을 RTOS 환경에서 통합하면서, 미리 약속한 CAN 통신 프로토콜에 의해 기능들을 구현하는 소스 코드를 만들었다.

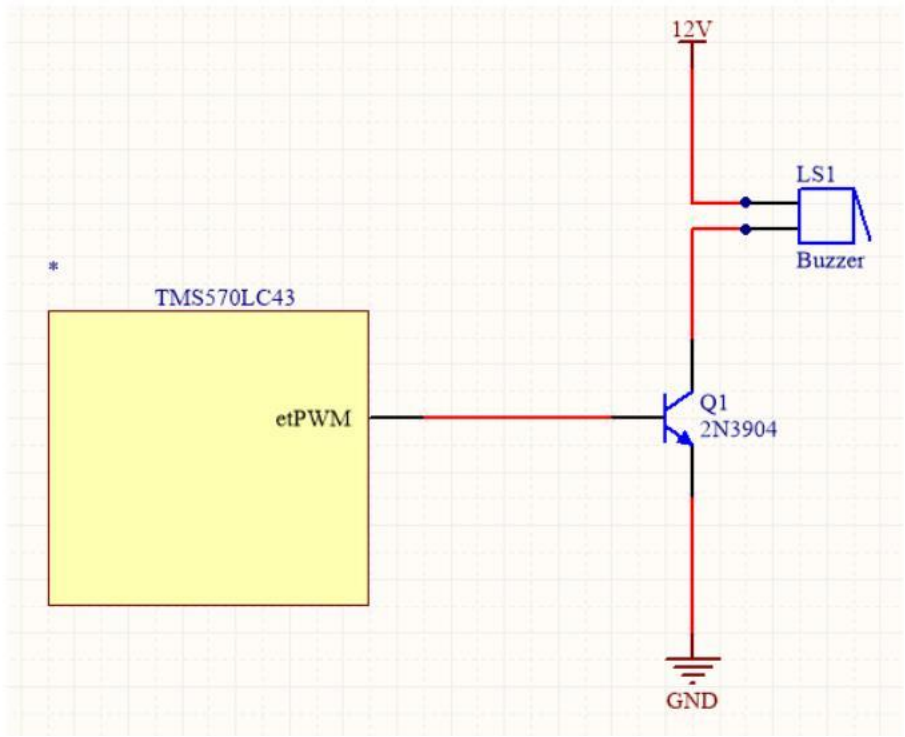
처음에 PC 와 MCU간에 TEST를 하기 위해 CAN 통신을 구현할 때는 PC에서 CAN 모듈을 통해 MCU로 전송하는 데이터 포맷을 알기 위해 UART 통신을 사용했다. 폴링방식에 의해 canGetData 함수를 사용해서 MCU에서 데이터를 수신하고, 데이터를 PC로부터 어떤 포맷으로 수신하는지를 확인하기 위해 UART의 sciSendByte 함수를 사용해서 터미널창에 데이터를 출력해 보았다. 이런 식으로 PC 로부터 MCU로 어떤 데이터 포맷으로 통신이 되는지 확인했고, 그로 인해 알게된 데이터 포맷에 맞게 PC와 MCU간의 TEST를 하였다.

하지만, 문제는 DSP 와 MCU간의 CAN 통신 데이터 포맷이 PC와 MCU간의 CAN 통신 데이터 포맷과 다르다는 것이다. DSP 와 MCU를 연결하고 DSP에서 CAN통신으로 데이터를 보내서 MCU를 제어하려는 TEST를 할 때, 처음에 DSP에서 아무리 프로토콜에 맞게 데이터를 전송해도 MCU가 동작하지 않았다. 처음 문제가 생겼을 때는 PC와 DSP가 CAN 통신 데이터 포맷이 다르다는 것을 알지 못하고, 계속 소스 코드만 들여다 보았다.
그 후, 소스코드에는 문제가 없다는 것을 깨닫고 DSP에서 MCU로 전송되는 CAN 통신 데이터 포맷을 UART 를 통해 PC의 터미널창에 출력해 보았다. 역시 PC와 TEST를 할 때와 데이터 포맷이 다르다는 것을 알게 되었고, DSP의 CAN 통신 데이터 포맷에 맞게 소스코드를 수정했다.

이러한 많은 시행착오를 겪고 DSP 와 MCU의 CAN 통신을 성공적으로 구현하였다.

3. 충돌 경보 사이렌, 상황별 LED 회로 제어

3-1. 충돌 경보 사이렌 회로



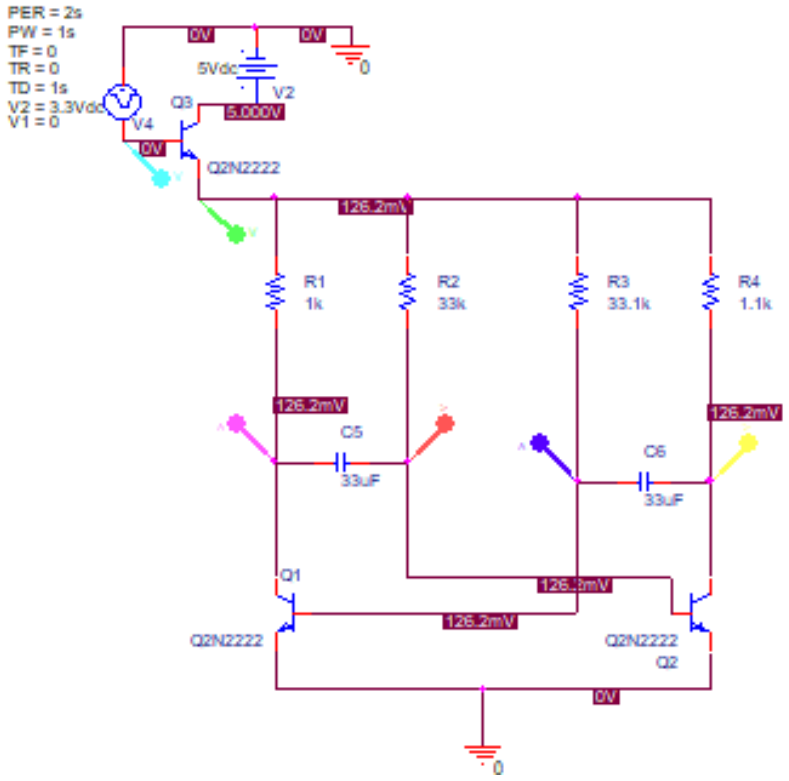
*** 구현 방법**

- MCU에서 버저를 직접 구동하기에는 전류량이 부족하다고 판단함.
- TR (2N3904) 소자를 오픈 컬렉터 회로로 구성하고 VCC는 MCU 전원인 12V를 연결함.
- TR의 컬렉터 부분에 가변저항을 달아서 버저의 소리 크기를 조절할 수 있게 만드려고 했으나, 버저의 소리 크기를 최대로 하기 위해 가변저항을 빼기로 함.
(버저안에 저항이 들어가 있기 때문에, TR의 컬렉터 부분에 저항을 달지 않아도 무관함)
- MCU의 etPWM 신호를 TR의 베이스에 인가함.

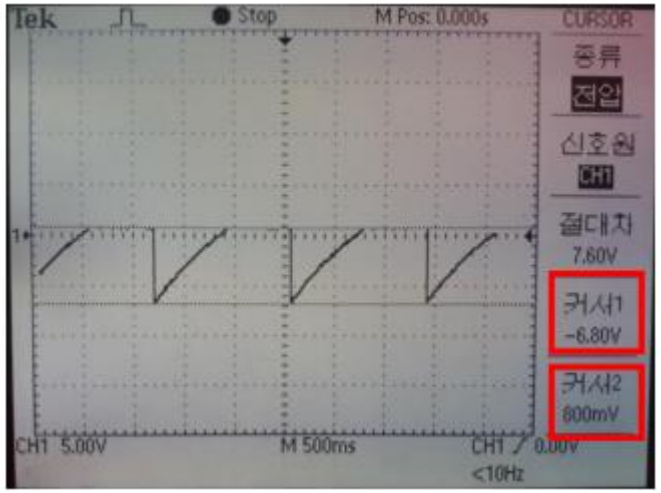
- etPWM 설정은 주파수 7HZ, duty비는 70%로 함.

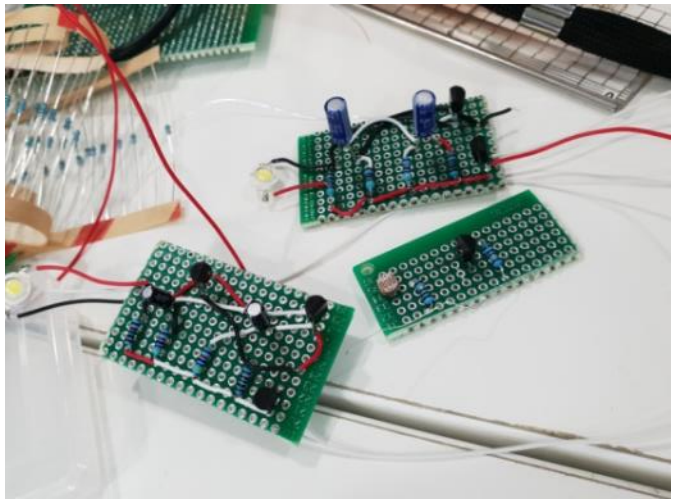
(duty비가 크면 소리가 울리고, duty비가 작으면 딱딱 끊어짐.)

3-2. 경보등(비안정 멀티바이브레이터)

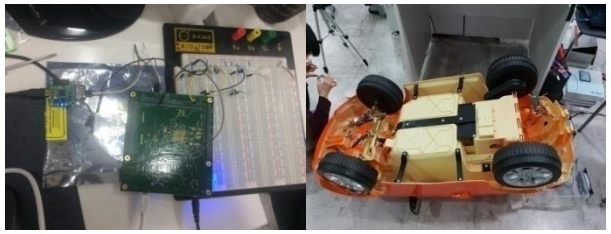


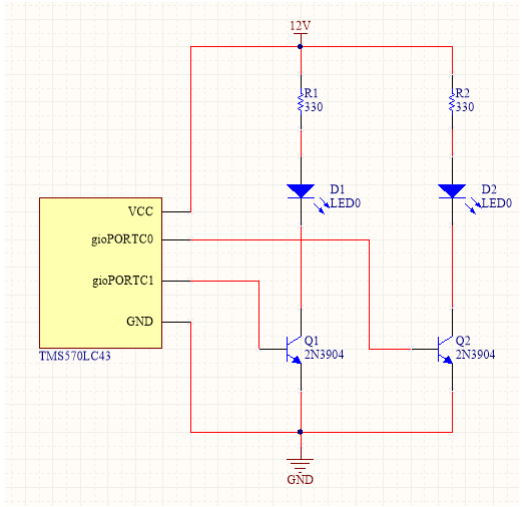
- 어떤 물체가 자동차에 일정 거리 이하로 가까워지면 Q3 TR의 베이스에 연결된 GPIO에 HIGH 신호를 인가해서 경보등을 동작시킴.





3-3. 방향지시등, 전조등





*** 목표: 차량의 방향지시등을 구현한다.**
야간, 실내와 같은 저조도 환경뿐만 아니라 일반적인 낮 환경에서도 선명히 보일 수 있도록 한다.

*** 구현방법**
6V로 동작하는 고휘도 LED를 사용하였다.
MCU 출력(3.3V)만으로는 원하는 성능을 얻을 수 없었으므로 OC회로를 구성하여 전류부족을 해결하였다.
방향지시등은 초기에는 RTI를 사용하여 GPIO Toggle, 했지만, RTOS상에서 RTI를 사용할 수 없었으므로 etPWM을 이용하여 구현하였다.
(주기 0.8sec Duty는 50% 사용)

*** 동작과정**
CAN으로 데이터를 입력받는다.
Can Data = 7이면 좌측, 8이면 우측 방향지시등을 작동시킨다.

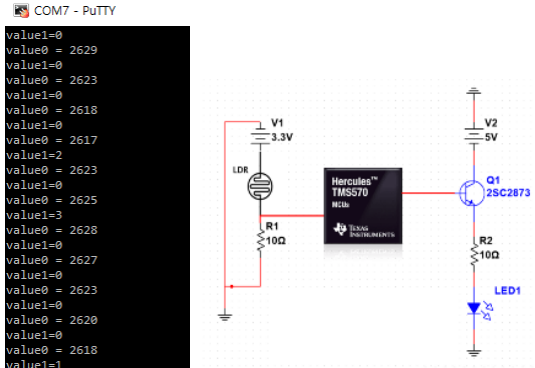
*** 목표: 차량의 전조등을 구현한다.**
임계밝기에서 자동으로 On/Off 되도록 한다.

*** 구현방법**
조도센서의 값을 ADC(Resolution 12bit)로 입력 받아서 실험적으로 원하는 임계값을 구했다.
임계값 이하에서는 전조등이 자동으로 켜지도록 하였다.
ADC Trigger 는 software적인 방법으로 구현하였다.
(GIOB 0번 핀 사용)
데이터 시트 상에서 다음과 같은 공식을 확인하고 입력전압에 따른 Digital Output을 예측해보았다.

$$DigitalResult = \frac{4096 \times (InputVoltage - AD_{REFLO})}{(AD_{REFHI} - AD_{REFLO})} - 0.5$$

실험 임계값 = 예측값
전조등은 OC로 구성하여 임계값 이하일때 TR의 Base로 GPIO신호가 입력되도록 하였다.

*** 동작과정**
ADC Trigger (GIOB[0]) 입력 한다.
임계값 이상이면 전조등을 Off, 이하이면 On 시킨다.



3-4. 고 찰

처음에 충돌 경보를 위해 사이렌 회로를 만들 때, 소리를 어느정도 크기로 하고 어떤 소리를 내야 할지 감을 못잡았다. 그래서 다양한 소리를 TEST해보기 위해 다양한 종류의 버저를 써보고, 스피커를 사용해 보기도 하고 소리의 크기를 위해서는 TR 스위칭 회로에서 TR의 컬렉터 부분에 가변저항을 연결해서 저항 값을 조정해가면서 소리의 크기를 TEST 해보았다. HALCoGen 툴에서 주파수와 duty비를 다양한 값으로 바꾸어 가면서 TEST 했다. 나 혼자 TEST 하는 것이 아니라, 팀원들과 같이 소리를 들어보면서 적절한 소리를 찾아 나갔다.

여러 소리를 들어보면서 결국 7HZ 주파수의 duty비 70프로의 etPWM이 가장 적절했고, 버저를 달고 저항은 달지 않는 것이 소리를 최대로 낼 수 있어서 좋다고 판단했다. 그 결과 충돌 경보를 울리는데 가장 적합한 사이렌 회로를 만들게 되었다.

4. LCD 제어

*** 사용 모듈**
LCD 모듈 : 1602A-1
I2C 인터페이스 모듈 : PCF8574

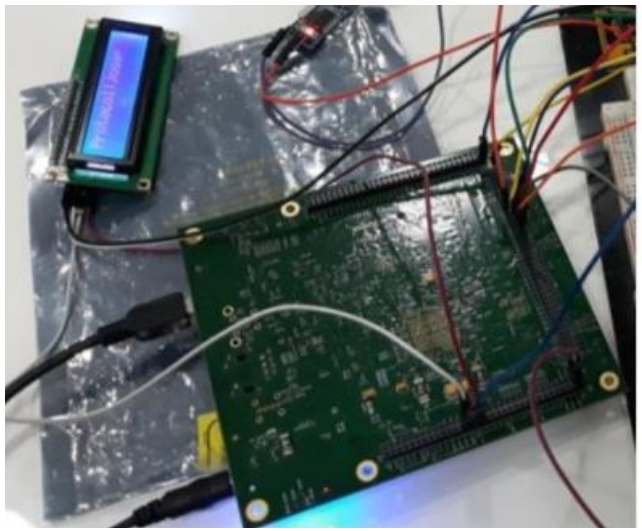
- 차 안에 총 2개의 LCD를 설치함.

- 하나의 LCD에는 BLDC(차 속도 제어) 모터의 속도 (km/h), 차의 좌/우회전 상태를 표시함.

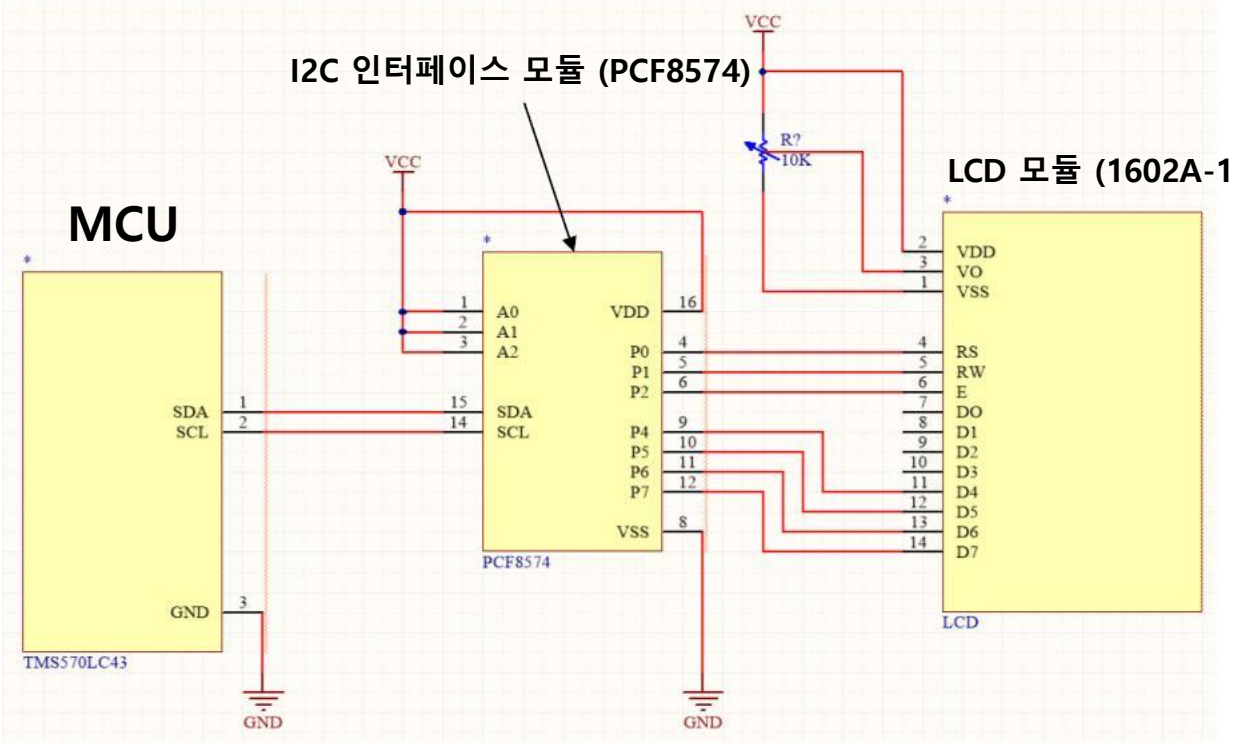
- 다른 LCD 에는 차 내부의 온습도 센서를 통해 얻어지는 현재의 온도, 습도를 나타냄.

- PCF8574A 모듈을 LCD에 연결해 I2C 통신으로 LCD를 제어함.

- I2C 통신이 속도가 느린 통신이고, LCD 액정에 글자를 띄우는 속도도 느려서, RTOS 환경에서 LCD를 제어하는 것이 굉장히 힘들었음.



4-1. LCD 제어 회로도



- MCU에서 I2C 통신 인터페이스로 LCD에 붙어 있는PCF8574 모듈을 제어하게 됨. (HW, SW적으로 매우 간단해짐)
- MCU로 LCD를 직접 연결하게 되면, MCU의 GIO 핀 8개를 사용해서 LCD를 제어해야 함. 이는 HW적으로 비용이 들게되고, 프로그래밍도 복잡해지므로 좋은 방법이 아님.
- I2C 통신은 HALCoGen 에서 제공하는 I2C 라이브러리 함수를 사용해서 구현함.

4-2. 고 찰

기존에 MCU와 LCD를 인터페이스하는 것은 PCF8574 와 같은 I2C 모듈을 사용하지 않고 MCU의 GPIO를 사용해서 LCD를 제어하는 경험을 해보았다. 하지만 그렇게 하면 프로그래밍이 복잡해지고, RTOS 환경에서 프로그래밍이 복잡해지면 스케줄링이 원할하지 않기 때문에 좋지 않았다. 또한 MCU의 GPIO 핀을 8개나 써야하기 때문에 다른 기능을 제어하는데 사용할 GPIO 핀이 모자르게 된다. 이러한 이유로 LCD의 I2C 인터페이스 전용 모듈인 PCF8574를 사용했고, HALCoGen 툴에서 제공하는 I2C 라이브러리 함수를 사용해서 프로그래밍을 굉장히 간단하게 구현했고, HW적으로도 SDA, SCL 2개의 핀을 사용하게 되어서 단순해졌다.

MCU와 PCF8574 모듈의 I2C 인터페이스를 소스코드로 구현할 때는 PCF8574 모듈의 데이터 시트를 참조했다. 그러나 데이터 시트를 보아도 MCU에서 어떻게 프로그래밍을 해야 할지 잘 이해가 가지 않았다. 인터넷 검색을 통해 다른 MCU로 작성한 소스코드가 있는지 찾아 보았고, 우연히 ST 시리즈의 MCU로 작성된 소스코드를 발견하였고, 해당 소스코드를 우리가 사용하는 MCU에 맞게 포팅했다. 소스코드가 그렇게 길지 않았기 때문에 포팅하는 것은 크게 어렵지 않았다.

소스코드를 포팅하고 기본적인 LCD 동작을 TEST해보았고, LCD의 문자 SHIFT 기능, 첫 문자 위치 정하는 기능 등을 구현해보았고 관련 명령어 데이터를 define 해서 우리가 쓸 LCD 라이브러리를 만들었다.

5. BLDC, 서보모터 및 EQEP 제어

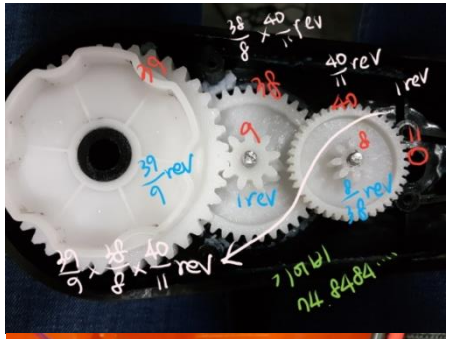
BLDC 모터

모터와 접속되는 기어

5-1. BLDC Sensored Motor + ESC를 이용한 속도제어

*** 사용 모듈**
- 1차 Turnigy XK-4074 2000KV Brushless Inrunner + HobbyKing® ™ X-Car Beast Series ESC 1:8 Scale 150A
- 2차 XERUN-150A-SD ESC + XERUN-4274SD 2200KV Sensored Motor

*** 구현방법**
BLDC 모터는 DC모터와는 다르게 내부에 장착된 홀 센서의 출력에 따라
3상 전류를 각각의 코일에 흘려주어야 작동된다.
따라서 직류를 BLDC 모터가 쓰기 위해 3극으로 변화시켜주는 ESC가 반드시 필요하다.
Sensored BLDC에서 나오는 A,B,C상 출력은 eQEP에 입력으로 넣어서
BLDC Motor의 회전속도를 구하는데 쓰인다.
Sensored BLDC의 분해능은 Motor Coupler를 사용하여
분해능을 알고 있는 엔코더 장착형 DC모터와 맞물려서 구했다.
PWM 20ms 주기의 5%, 10%를 각각 min, max duty로 설정하여
For/Rev 모드의 ESC Calibration을 진행한다.
Calibration이 끝나면 원하는 Duty를 주어 모터를 정/역회전시킬 수 있다.



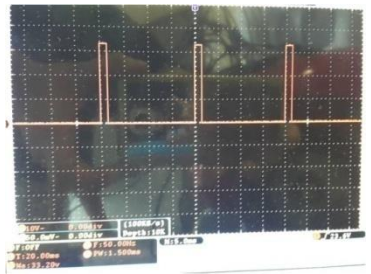
일반적인 RC Car에서 사용하는 Duty와는 오차가 있으므로 duty를
50간격으로 변화시키면서
모터를 구동시킬 수 있는 duty를 구해보았다.
차량에 있는 기어박스의 기어비가 약 1: 74 이므로
차량의 바퀴는 모터 회전속도의 1/74배 모터 토크의 74배 힘을 낼 수 있다.

*** Issue사항**
모터와 변속기는 2회 구매하였다.
1회 구매시 Sensored Motor가 아니었기 때문에
외부에 엔코더를 부착하는 방식과
Sensored 모터를 구매하는 방식의 두가지를 놓고 고민하였는데
엔코더 부착의 기구적인 어려움으로 인하여 후자를 택하였다.

피니언 기어도 다음과 같이 2회 구매하였다.
10T/5mm M1 Hardened Steel Pinion Gear (1pc)
Hobbywing 11T 5mm M1 STEEL PINION GEAR

*** 동작과정**
etpwmREG2의 TBPRD와 COMPB값을 조정하여
모터에 인가하는 PWM Duty를 조정할 수 있다.

Stop
5% 최대 역최전 ⟷ 10% 최대 정회전

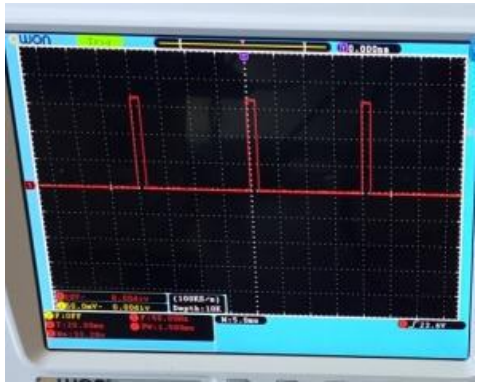


```
/** - Sets time period or frequency for E
etpwmREG2->TBPRD = 19999U;

/** - Setup the duty cycle for PWMA */
etpwmREG2->CMPA = 10000U;

/** - Setup the duty cycle for PWMB */
etpwmREG2->CMPB = 1000U;
```

5-2. Servo Motor를 이용한 조향제어

*** 사용 모듈**
- 1차 TrackStar™ TS-910 Digital 1/8 Truggy/Monster Truck Servo 30.6kg / 0.14sec / 66g
- 2차 KINGMAX DCS16942CHV 186g 50kg.cm metal gear big size Servo

*** 구현방법**
Servo Motor는 주어진 PWM의 Duty비에 따라 일정한 각도를 유지하는 모터이다.
따라서 정확한 방향제어를 위해서 기존에 활용되던 DC모터를 대체하여 사용하였다.
기존에 DC모터를 이용하여 조향에 활용되던 기어박스를 그대로 재활용하기 위해서
기존 차량의 기어와 모듈비가 일치하는 기어를 청계천에서 구매하였다.
20ms에 5%~10%의 duty비를 입력하면 0~180도 제어가 가능할 것이라고 생각하였으나,
실제로는 0.5ms와 2.5ms 의 duty비를 입력하여 제어할 수 있었다. 이는 히스테리시스
때문이다.



5-3. 엔코더 출력을 eQEP입력으로 받아서 모터 속도 출력

*** 구현목표 :** Sensored BLDC의 엔코더 출력을 이용하여, 모터의 현재 속도를 출력한다.
이를 PID 제어에 활용한다.

*** 구현방법**
Sensored BLDC의 엔코더 성능에 대한 정보가 없었으므로
방법 1로 분해능을 알고있는 엔코더를 사용하여 BLDC의 엔코더 PPR을 구한다
일반적인 값과 차이가 많이 나서 확인을 위해 방법 2를 사용하였고
2PPR임을 최종적으로 확인하였다.

방법 1. 432ppr의 주기와 BLDC의 주기를 비교하였다.

방법 2. BLDC를 수동으로 한 바퀴 돌렸을때 엔코더의 출력파형을 관찰하였다.

TMS570LC43의 eQEP는 고속, 저속의 두가지 측정모드를 가진다.
저속모드는 펄스간 시간을 측정하여 속도를 구하는 방식이고,
고속모드는 정해진 샘플링 주기 내에 몇 개의 펄스가 포함되는지를 이용하여 속도를 구한다.
프로젝트 환경에서 고속모드가 적합하다고 판단하였다.
따라서 샘플링 주기를 임의로 0.5초로 설정하고,
주어진 샘플링 주기 내에 들어오는 QCLK를 카운트 하여 QPOSLAT에 래치하게 된다.
속도는 QPOSLAT의 값을 이용하여 구할 수 있다.
BLDC Motor의 분해능은 2PPR이다. 따라서 1회전당 8개의 QCLK가 출력된다.
(1개의 QCLK당 360/8=45도 회전)
즉, QPOSLAT * 45도 / 샘플링타임(0.5sec)로 계산하면 BLDC Motor의 각속도를 계산할 수 있다.

바퀴의 각속도는 앞에서 구한 BLDC의 각속도를 기어비(약 1:74) 74로 나누어주면 구할 수 있다.
그리고 이를 360으로 나누어주면 바퀴의 초당 회전수(rps)를 구할 수 있다.

자동차의 속력(cm/sec)를 구하기 위해 바퀴의 지름을 측정하면 22.83cm이고,
1회전당 약 22.83 x 3.14 = 71.72cm 를 이동한다는 것을 알 수있다.
따라서 앞에서 구한 초당 회전수 x 1회전당 이동거리 = 초당 이동거리 즉 cm/sec를 구할 수 있다.

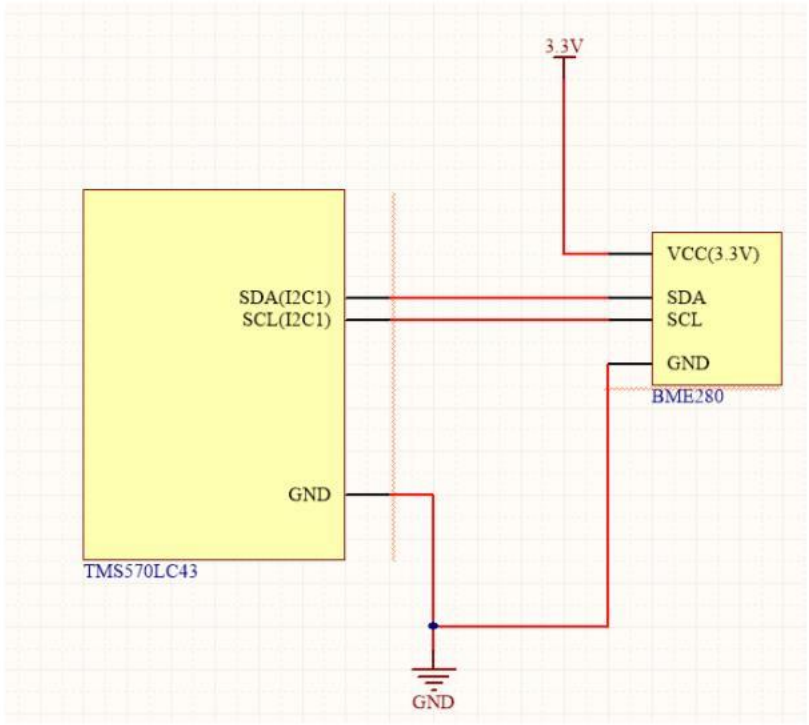
6. 온,습도 센서 제어

*** 사용모듈 : BME280**

- BLDC 모터가 빠르게 구동되면 발열이 심하게 남. 너무 심한 발열이 지속적으로 나게 되면 모터에 무리가 가게 되고 주변 기기에도 영향을 줄 수 있음

차 후방의 BLDC 모터가 있는 공간의 온,습도를 측정하기 위해 BME280 온,습도 센서 모듈을 사용함.

- I2C 통신 인터페이스를 사용함.



6-1. 고 찰

처음에 MCU와 BME280 모듈을 인터페이스 하기 위해서 데이터 시트를 보았다. 데이터 시트를 보니 BME280 모듈의 핀 중에서 SDO 핀이 존재했고, SDO핀은 SPI 통신의 MISO핀을 의미했다. 또한 I2C 통신으로 사용하는 SDA 핀은 SPI 통신의 MOSI 핀으로 사용이 가능했다. 즉, SPI 통신과 I2C 통신 2가지 통신으로 MCU와 인터페이스 할 수 있는 모듈이라는 것을 알게 되었고, 어떤 통신으로 인터페이스를 할 지 고민했다.
SPI 통신은 SS, SCL, MISO, MOSI 4개 핀으로 통신을 하는 4선식 동기식이라 HW가 복잡해진다. 하지만, I2C 통신은 SCL, SDA 2개 핀으로만 통신을 하는 2선식 동기식이기 때문에 HW적으로 매우 간단하다. 또한 일반적으로 I2C 통신을 사용하는 경우가 SPI 통신보다 빈도수가 많다는 점을 고려할 때, I2C 통신으로 인터페이스를 하는 것이 좋다고 판단했다.

I2C 통신으로 모듈 인터페이스를 하려고 할 때, 데이터 시트를 많이 읽어 보았지만 처음에는 감이 잘 안왔다. 센서이기 때문에 무작정 온,습도 데이터를 읽기만 하면 될 줄 알았다. 그래서 어떻게 해야 할지 감을 잡기 위해서 아두이노 라이브러리로 되어 있는 BME280 모듈의 인터페이스를 하는 소스 코드를 참조 했다. 아두이노 코드를 보니 무작정 데이터를 읽어야만 하는 것이 아니라 모듈 내의 각종 레지스터를 초기화를 시켜줘야 된다는 것을 알게 되었다.

아두이노 코드를 보면서 어떤 식으로 BME280 모듈과 통신하는지 감을 잡고 소스 코드를 설계해나갔다.

7. FreeRTOS 통합

*** 구현 방법**

- LCD, 충돌 경보 사이렌, BLDC 모터 제어등 내가 만든 기능들과 다른 팀원들이 만든 기능들을 RTOS 환경에서 통합하는 작업을 수행함.

- 비슷한 기능을 하는 것끼리 묶어서 태스크 별로 정리함. ex) TASK1 : 모터제어, TASK2 – LED 관련 제어

- 각각의 기능들이 main 문에서 단독으로 수행되는 경우에는 잘 되었는데 RTOS 환경에서 스케줄링에 의해 태스크에 의해 수행되니

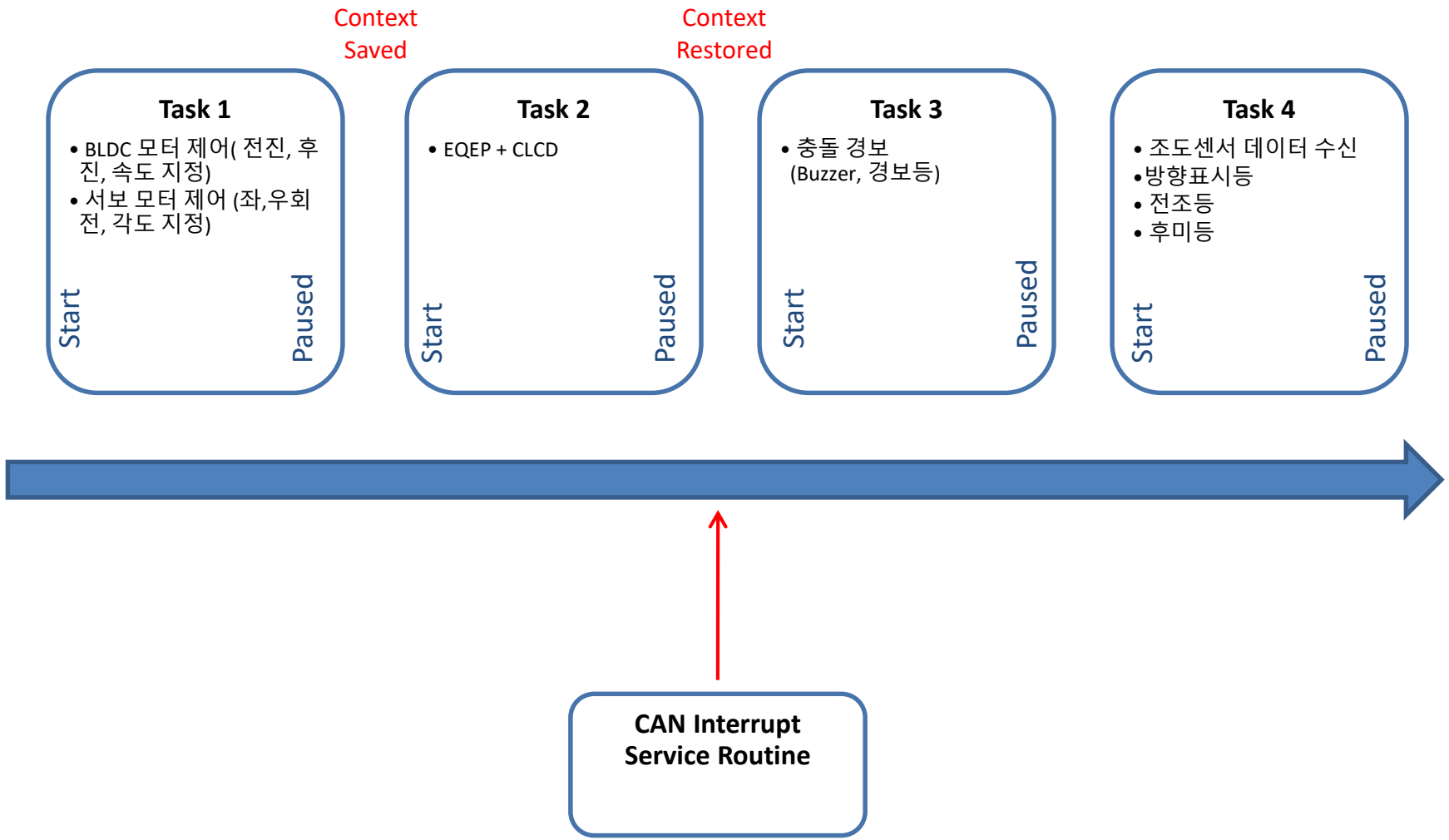
실행 속도가 느려지거나, 때로는 동작하지 않는 경우가 빈번하게 발생함.

- RTOS 환경에서는 무의미하게 딜레이 되는 구간을 최소화 시켜야 하기 때문에 기존에 폴링방식으로 수행되던 기능들을

모두 인터럽트 방식으로 변경함.

- CAN 통신으로 데이터를 수신하는 과정에서 태스크의 스케줄링이 일어나면 잘못된 데이터가 수신될 수 있으므로, 데이터를 수신하는 부분 전에 세마포어를 획득하게 하고, 데이터 수신이 끝나면 세마포어를 반납하게함.

7-1. FreeRTOS 기반 Flow Diagram



7-2. 고찰

팀원들이 각자 만든 부분의 프로젝트를 알집으로 압축해서 나에게 넘겨주면, 그것을 RTOS 환경에서 통합하는 작업을 수행했다.

팀원들이 나에게 소스 코드를 넘겨줄때, 기능별로 함수를 만들지 않고, main 문 안에서 모든 소스코드를 구현해버려서 그것을 내가 기능별로 함수를 분리하는 작업을 해야했다. 이러한 작업이 시간을 낭비하게 만들었고, 정말 귀찮은 작업이었다. 이러한 문제로 팀원들과 협업을 하면서 종종 마찰이 있었지만, 프로그래밍 능력이 떨어지는 팀원의 경우에는 하는 수 없이 내가 다시 소스 코드를 정리하는 작업을 하기로 했다.

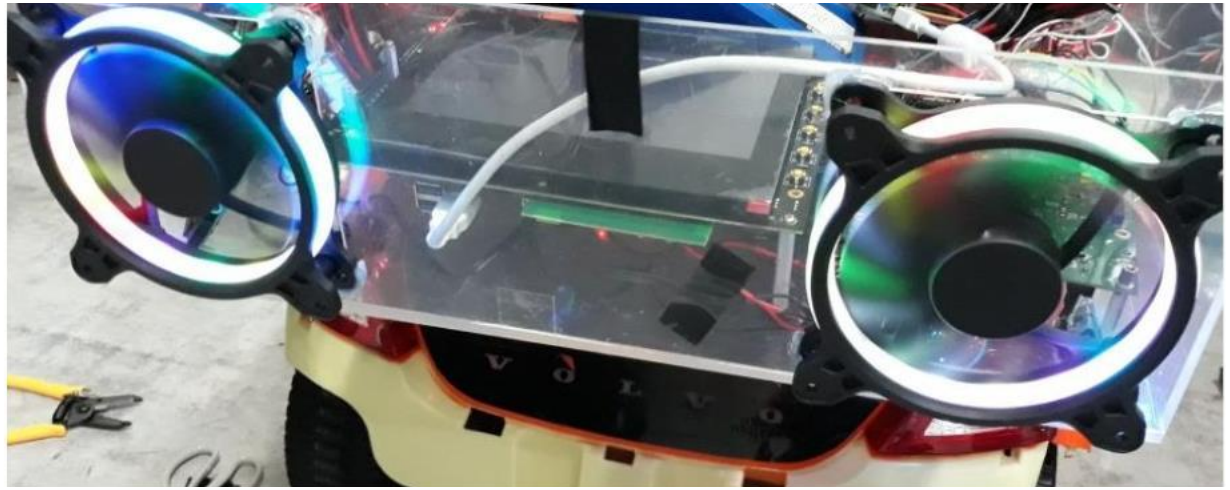
혼자 개발할 때도 마찬가지이지만, 여러사람이 협업을 해서 프로젝트를 하는 경우라면 더더욱 누구나 알아보기 쉽게 기능별로 함수를 만드는 것이 반드시 필요하다는 것을 깨달았다.

8. 쿨러팬 구동

- 아크릴 상자안의 DSP, FPGS, MCU 보드들이 발열이 심하기 때문에 열을 식히는 것이 중요함.

- 각각의 보드에 방열판을 달아서 열을 식힐수 있지만, 방열판으로는 턱없이 부족해서 쿨러팬을 설치해서 구동시키기로 함.

- 아크릴판 내부의 온도가 일정 수치 이하가 되면, 쿨러팬을 구동시키려고 했지만, 보드가 항상 뜨겁기 때문에 계속 구동시키기로 함.



- 일반적으로 모터를 구동하려면 모터 드라이브 전용 IC를 사용하거나, 그림 a와 같은 회로를 구성해서

모터를 구동해야 한다. 하지만 본 프로젝트에서 사용한 TMS570LC43 MCU의 전원에서 출력되는 전류는

1A이므로 모터를 구동하기에 충분하다.

따라서 별도의 회로를 구성하지 않고, 그림 a와 같이 간단하게 회로를 구성하였다.

- 쿨러팬을 구동하기 위해서 SW적으로 소스 코드를 구현하지 않고, HW 에 의해서만 쿨러팬이 구동되므로

RTOS 환경의 태스크 스케줄링에 전혀 영향을 주지 않는다.

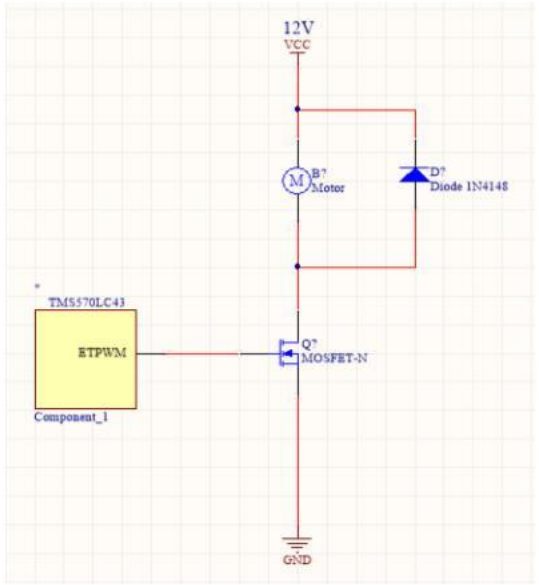


그림 a. FET를 사용한 모터 구동회로

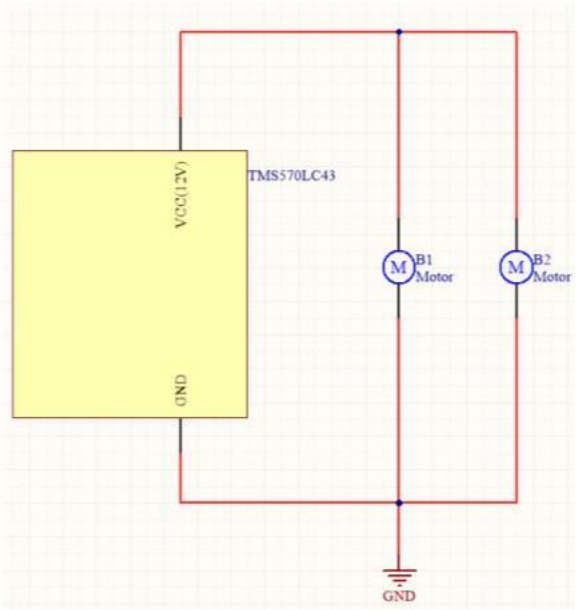


그림 b. 쿨러팬 구동 회로(프로젝트)

9. 디버깅 작업 (SCI)

=> 일반적으로 **UART** 라고 하지만 TMS570 MCU에서는 **SCI** (Serial Communication Interface) 라는 명칭을 사용한다.

CAN 통신, RTOS 통합 작업, BME280 모듈을 인터페이스 할 때 시행착오를 많이 겪었다. 그리고 CAN, I2C 통신같은 경우에는 한 번에 성공하지 못하고 계속 실패를 거듭했다. 그러면서 자연스럽게 디버깅을 많이 하게 되었다.

나만의 디버깅 방법은 의심이 가는 코드의 뒷부분에 SCI를 이용해서 터미널 창에 데이터를 출력하는 코드를 넣어 보는 것이다. 해당 데이터가 터미널 창에 잘 출력이 된다면 바로 앞줄의 코드는 정상적으로 동작한다고 볼 수 있다. 터미널 창에 데이터가 출력이 되지 않는다면 앞부분의 코드에서 문제가 생긴 것이다. 이러한 방식으로 문제가 있다고 의심되는 범위를 좁혀 나갔다.

터미널 창에 데이터를 출력하는 방법에는 printf 함수 사용과 sciSend 함수(uart통신의 송신함수) 사용의 2가지 방법이 있다. 2가지 방법을 모두 사용해본 결과 printf 함수는 다른 코드에 영향을 줄 정도로 속도가 느렸고, sciSend 함수는 디버깅 용도에 적합할 정도로 빠른 속도를 보여주었다.

또한 printf 함수는 RTOS 환경에서는 사용할 수가 없었다. 그렇기 때문에 printf 함수 보다는 UART 통신을 사용하는 것이 디버깅에는 적합하다고 생각한다.

10. DC-DC 컨버터

- MCU, DSP, FPGA 보드로 각각의 기능들을 구현하면서 전원을 AC-DC 어댑터를 사용했지만 차 안에 보드를 탑재시키려면 어댑터를 더이상 사용할 수 없게됨.

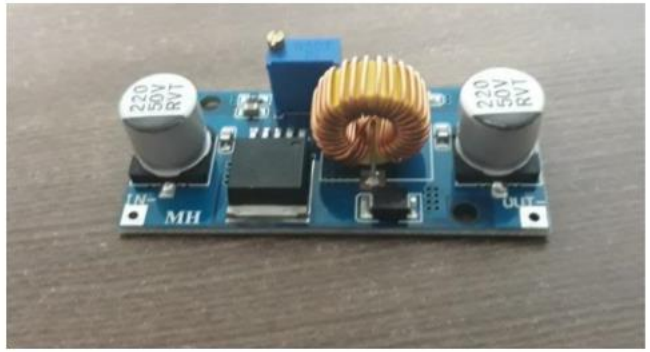
- DSP, FPGA 보드는 시중의 DC-DC 컨버터 모듈을 사용하고 MCU의 전원부에 연결할 DC-DC 컨버터 모듈을 만들기로 함.

- 사용하는 MCU 보드의 전원 스펙은 12V, 1A임.

- 22.5V Li-Po 6S Battery를 전원 공급원으로 사용하고 있으므로, 22.5V를 12V로 낮춰줄 벅 컨버터가 필요함.

- DC-DC 컨버터 에 대한 지식이 전무해서, 팀원들끼리 스터디를 해서 먼저 이론을 공부하기로 함.

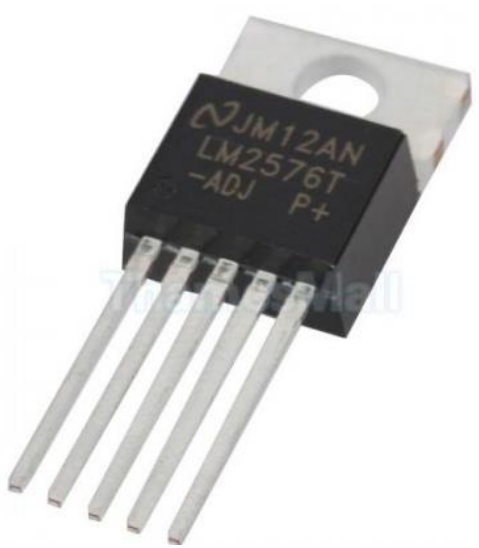
- 개발할 벅 컨버터에 대한 회로 및 IC, 소자를 선정하고 IC의 데이터 시트를 통해 원리를 분석함.



10-1. IC 선정

12V 1A의 벅 컨버터를 설계하기 위한 IC로 TL494, IR2110 을 선정했다. 2가지의 IC로 DC-DC 컨버터를 설계함.

- TL494 : PWM 파형 발생용 IC
- LM2576-adj : DCDC 벅 컨버터를 설계를 위한 단일 IC



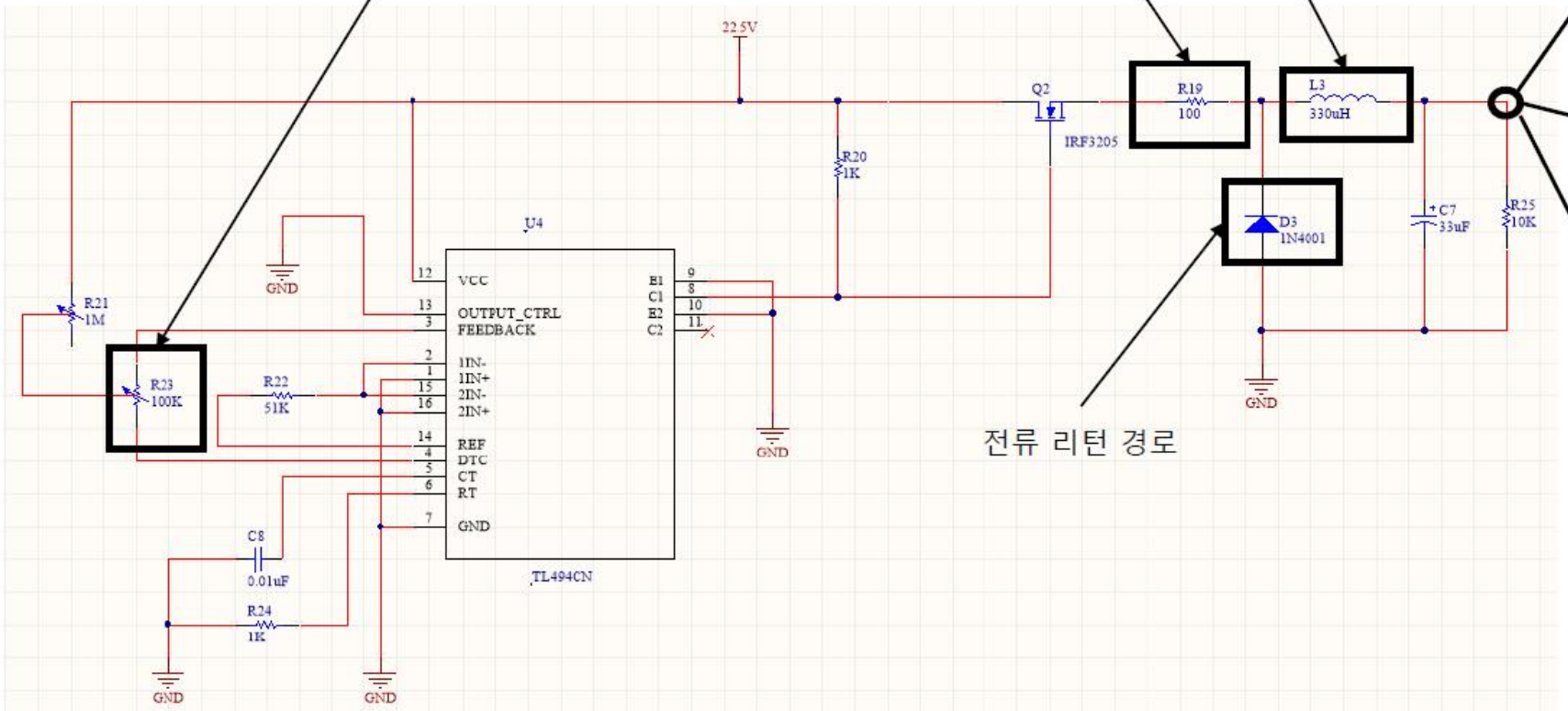
10-2. DC-DC 컨버터 설계(TL494)

TL494 의 출력 PWM 파형의 Duty 비를 조절해서
출력 전압(Vout) 을 조절 가능!

출력 MAX 전류 결정

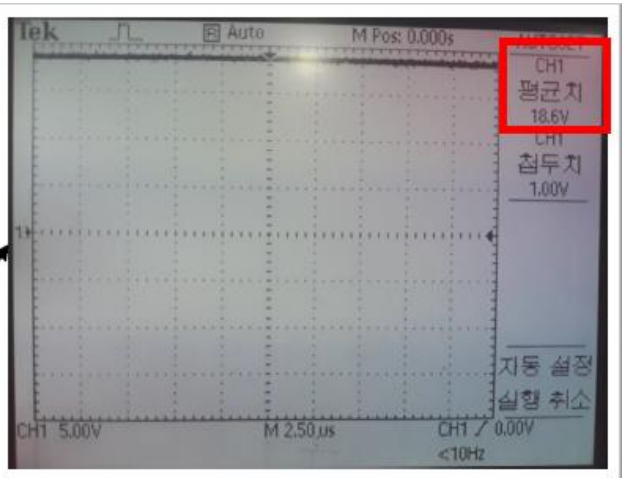
전류 제한용 저항

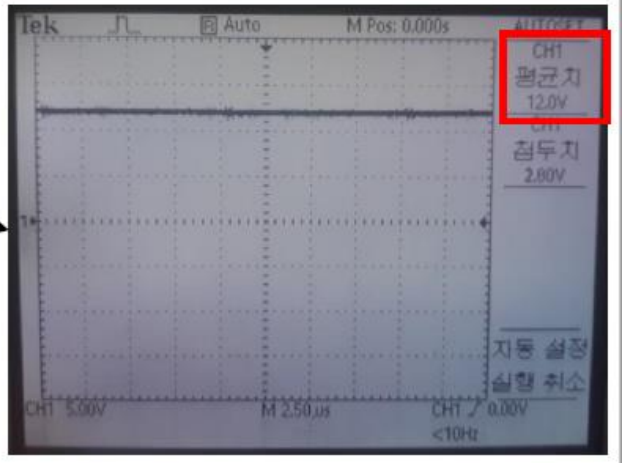
전류 리턴 경로

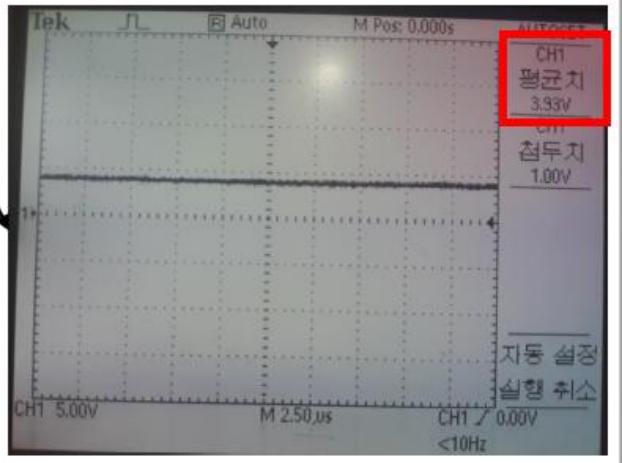


R23 가변저항으로
출력전압 조절 가능!

최고

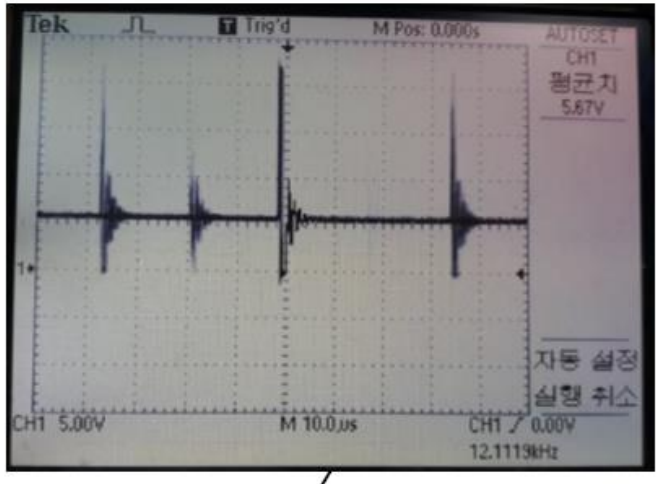






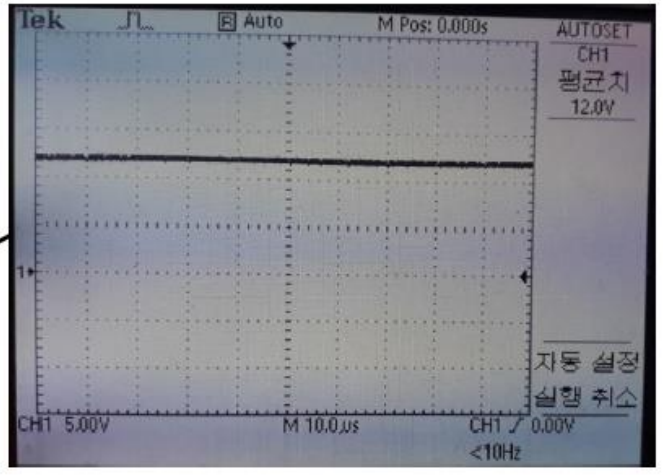
최저

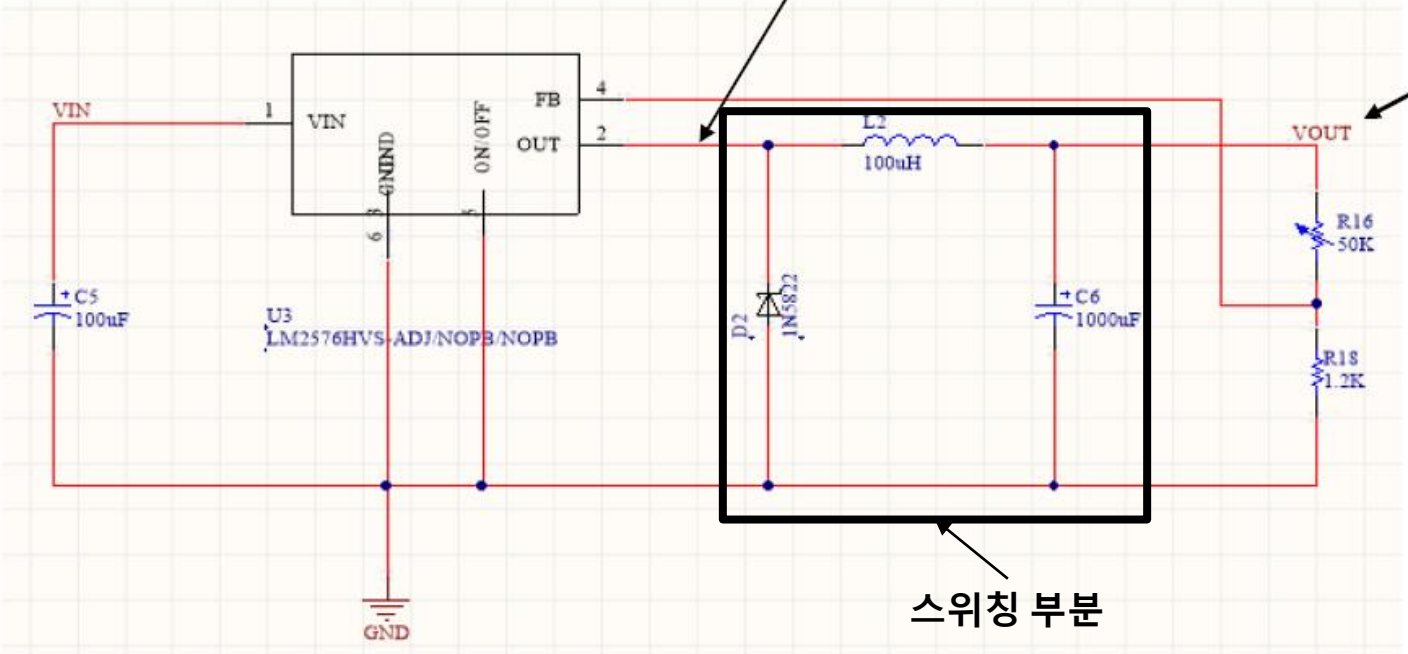
10-3. DC-DC 컨버터 설계(LM2576-ADJ)



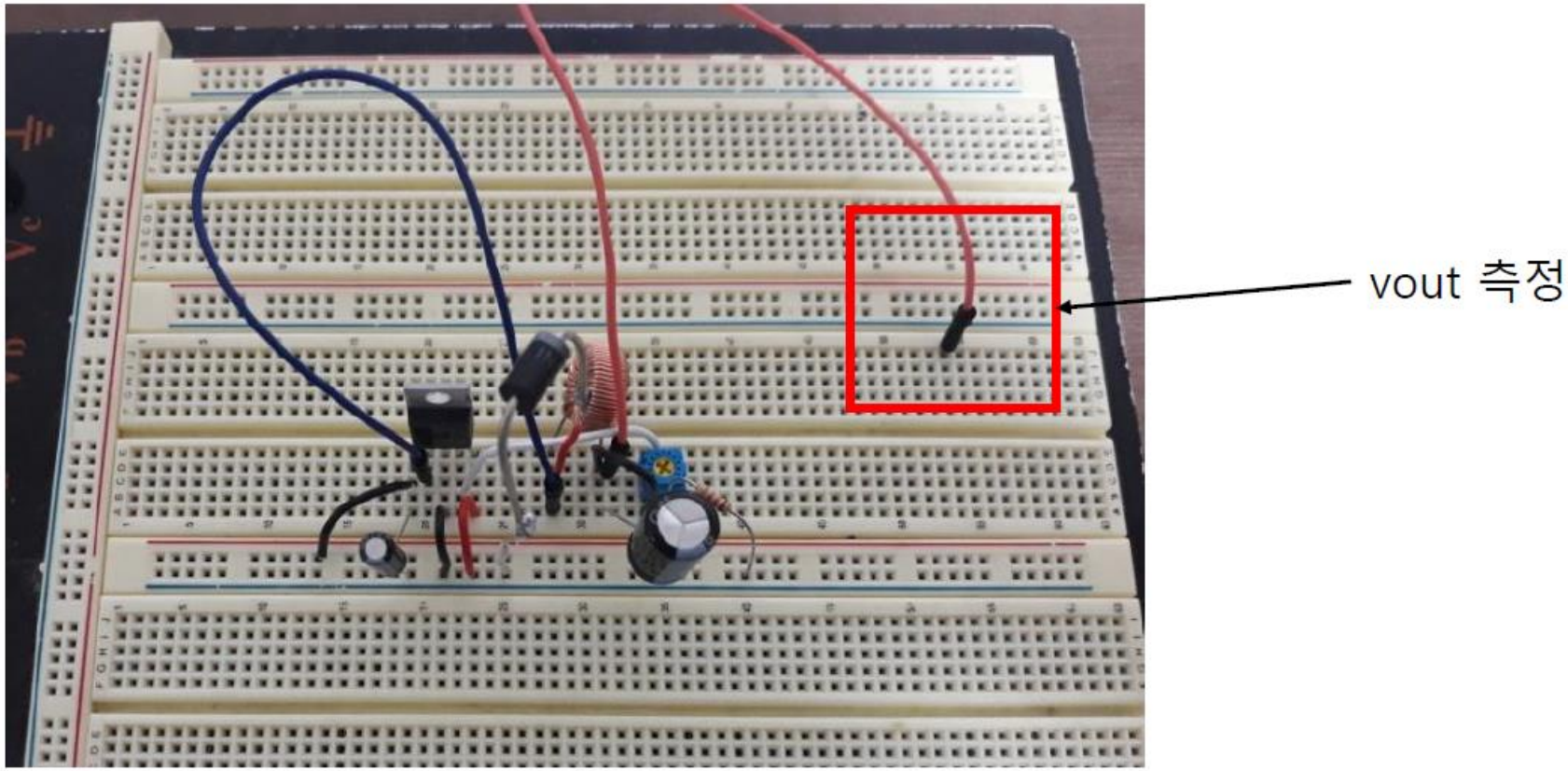
R16 값에 따라 DC 성분이 바뀜!

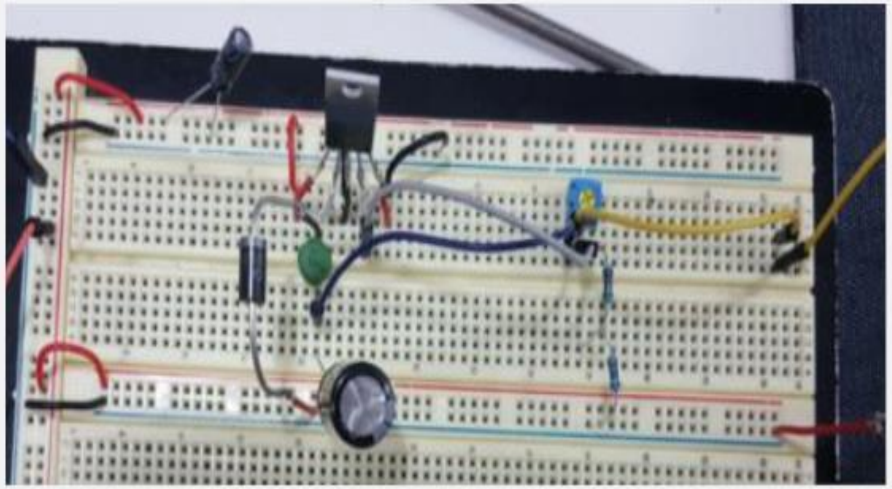
R16 값을 조절해서 12V를 맞춤!

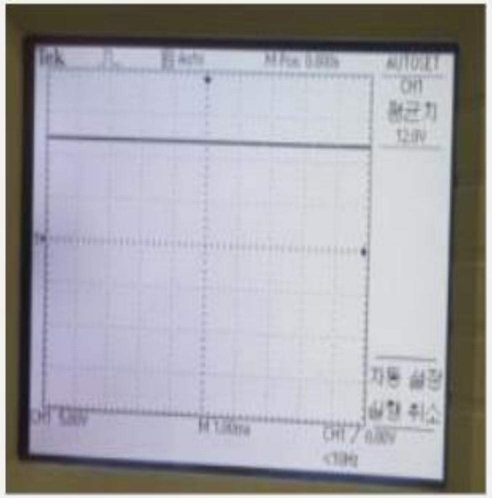




10-4. 브레드 보드 기반의 TEST 모습







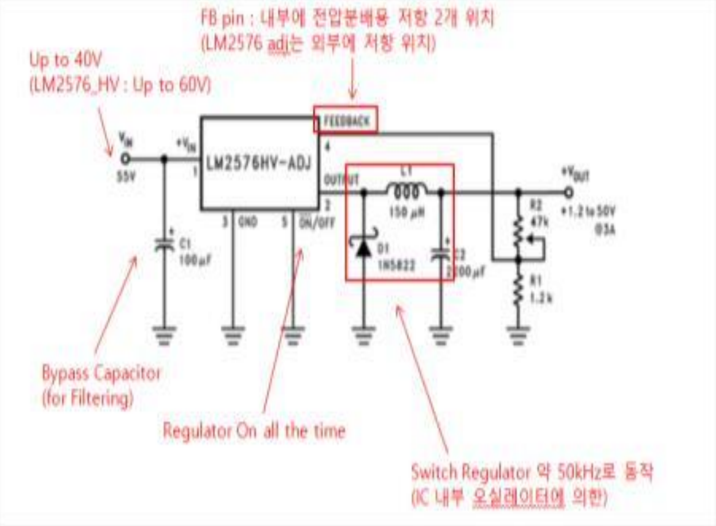
견 적 서

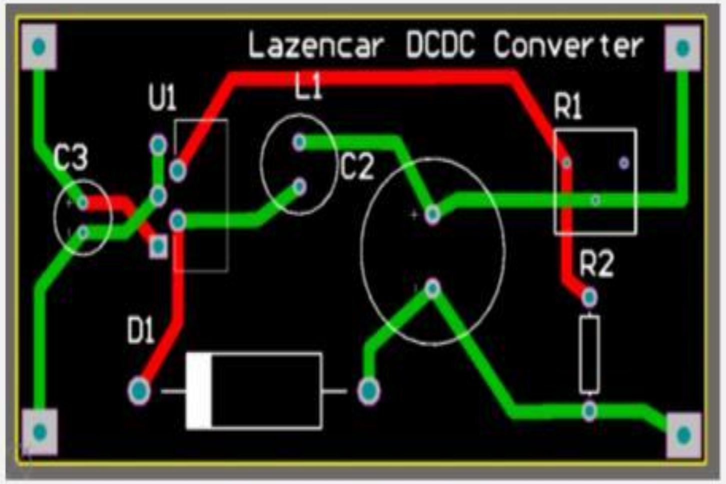
날짜:2018 년 09 월 28 일

품 명 및 규 격	수량	단가	공급가액	세 액	비 고
PCB Project_Lazenc 060 x 030 1.6T 2L 2D	4	0	54,000	5,400	
Project_Lazen 060 x 030 1.6T 2L 2D		0	20,000	2,000	
합 계			74,000	7,400	

<기타 참고 사항>

재 질	FR-4
결제방법	
결제계좌	신한은행: 140-004-083846
데이터 전송	E-mail: order@hsdgt.com





11. 프로젝트 진행 중에 느낀점

1. 어떠한 MCU라도 원리는 비슷하고, 라이브러리를 사용하면 편리하고 빠르게 개발할 수 있다!

=> 처음에 TMS570 이라는 생소한 MCU를 접했을 때 굉장히 어려울 것이라고 생각했다. 나는 이전에 AVR, STM32 같은 MCU만 경험했기 때문이다. 하지만 막상 TMS570 을 다루어 보니 기존에 경험했던 MCU와 크게 다르지 않다는 것을 깨달았고, peripheral 또한 이름만 다를 뿐 원리는 같았다. 레지스터에 직접 접근을 하며 개발을 하는 것보다는 코드 자동생성 툴을 이용해서 라이브러리를 사용하는 것이 더 빠르다고 생각했다.

2. 좋은 엔지니어가 되기 위해서는 강한 정신력이 필요하다!

=> MCU와 BME280 모듈을 인터페이스하면서 굉장히 어려움을 많이 겪었다. I2C 통신 외에 SPI 통신을 공부하기 위해 SPI 통신으로 인터페이스를 했다. SPI 통신을 GPIO로 구현한 아두이노 소스 코드를 우리가 쓰는 MCU로 그대로 포팅해서 구현을 해보았지만, 계속 통신이 되지 않았다.

모든 소스 코드를 약 한달 동안 계속 검토했지만 원인을 찾지 못했다. 의심이 가는 코드 앞 뒤를 UART를 통해 터미널창에 데이터를 출력해서 디버깅을 수도 없이 했다. 아무 이상이 없는데 전혀 온도 데이터가 나올 기미가 없었다. 약 한달 동안 BME280 모듈과 씨름을 하면서 정신적으로 굉장히 괴로웠다. 같은 소스 코드를 계속 들여다 보는 고통은 느껴본 사람만 알 것 같다. 결국 SPI 통신의 속도 문제가 원인임을 밝혀 냈다.

3. 여러사람과 협업을 해서 프로젝트를 할 때에는 확실한 역할 분담이 필수적이다!

=> 프로젝트 초기에 팀원들의 역할 분담이 제대로 되지 않다 보니 하는 일이 중복되는 경우가 잦았고, 그러다 보니 프로젝트 진도가 잘 나가지 않았다. 확실하게 팀원 간에 역할분담을 하고, 어려운 부분만 같이 해결해 나가는 방법이 올바른 방법이라고 생각한다.

12. 프로젝트 결과

주행 테스트 영상: https://www.youtube.com/watch?v=jwrWfwDbgT0&feature=youtu.be