전기전자 엔지니어를 위한 노이즈대책 실무 경험담

노이즈는 내친구

본 책은 컴파일 공식 블로그인 http://cubloc.blog.me 에 있는 내용중 제품 신뢰성을 높이는 데 도움이 될 만한 자료를 모은 것입니다. 모쪼록 고객 여러분의 실무에 도움이 되었으면 합니다.

컴파일 테크놀로지 주식회사 대표이사

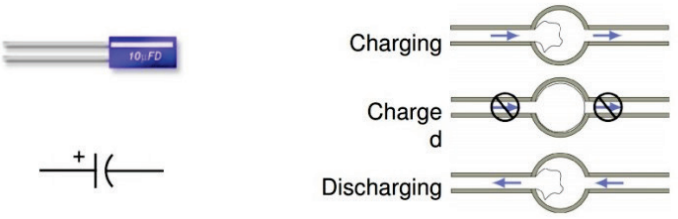
목차

제 1 장. 기초소자탐구

노이즈 대책에 사용되는 소자입니다.

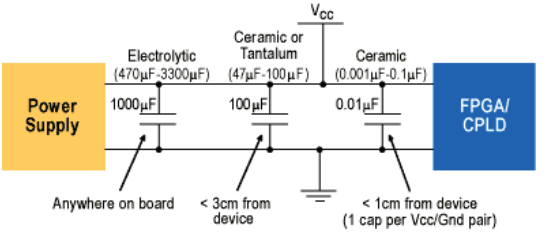
콘덴서

콘덴서는 전기를 (아주 잠깐동안) 저장하는 소자입니다.



위 그림을 보시면, 전기를 저장하는 시간 (Charging)동안은 전기가 흐릅니다. 전기가 다 저장되면, 더이상 전기는 흐르지 않습니다. 전기가 방전 (Discharging) 되는 동안 반대방향으로 전기가 흐릅니다.

콘덴서는 노이즈 대책에 있어서 핵심적인 역할을 하는 부품입니다. 전기를 잠깐 저장한다는 특성을 노이즈 대책에 적극 활용하고 있는 건데요. 시중에 유통되고 있는 보드를 보면 전원회로에서부터 칩 주변까지 콘덴서로 도배가 되어 있다시피 합니다.

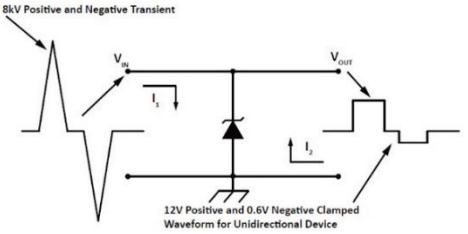


각각의 콘덴서마다 충분한 이유가 있어서 그 자리를 차지하고 있는 것이죠. 그중 IC 마다 붙어있는 콘덴서는 노이즈 감소에 매우 중요한 역할을 수행합니다. (뒤에서 자세히 다룸.)

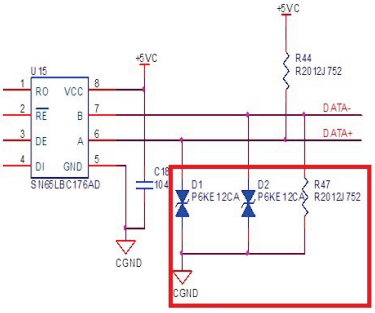
TVS 다이오드

여러분이 악수하다가 정전기를 경험하기도 하시죠? 전자 제품도 마찬가지로 접촉이 빈번한 부위에 예상치 못한 높은 전압이 들어와서 부품이 망가지는 일이 발생합니다.

특히, RS232 콘넥터나 RS485 콘넥터, USB 콘넥터등은 사람들이 수시로 케이블을 꼽았다, 뺐다 하는 곳입니다. 이 회로의 콘넥터와 소자 사이에 TVS 다이오드를 부착하면, 소자가 파손되는 것을 막아줍니다.



그림처럼 입력 전압이 뾰족하게 치솟아도, TVS 다이오드를 통과하면 일정 전압이상은 깍이게 됩니다. 다음은 RS485 회로에 TVS 다이오드를 부착한 예입니다.

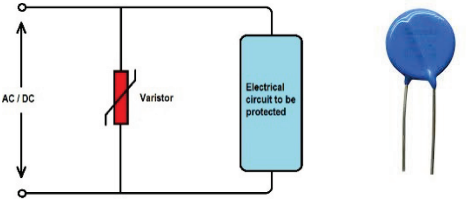


주의하실 부분은 TVS 가 순간적인 과전압 (수십 ns 정도)에 대한 보호만 해준다는 것입니다. 지속적인 과전압은 보호할 수 없습니다. TVS 다이오드를 적재적소에 사용해준다면, 여러분이 만든 보드를 A/S 하러 다니는 일이 현저하게 줄어들 것입니다.

바리스터

바리스터는 가해진 전압이 증가하면 저항 값이 크게 줄어드는 성질을 가진 반도체 소자입니다. 한마디로 일정 전압이 흐르는 회로에서 보호해야하는 소자가 있는 곳에 병렬로 연결해 놓으면, 갑자기 큰 전압이 흐를 경우, 바리스터의 저항이 확 낮아지기 때문에 보호할 소자쪽으로 전류가 흐리지 않고 바리스터쪽으로 전류가 흐릅니다.

특히, 서지 (갑자기 큰 전압이 일시적으로 들어오는 현상)가 발생하는 곳의 서지흡수에 탁월한 성능을 발휘합니다.



TVS 다이오드와 사용하는 목적은 비슷합니다만, 바리스터는 마치 콘덴서처럼 정전용량을 가지고 있어서 고속신호가 왔다 갔다 하는 곳에서는 사용하지 않는 것이 좋습니다. 신호가 뭉그러지는 현상이 생깁니다. 고속신호 (예를 들면 I2C 통신)를 쓰는 곳에서는 TVS 다이오드를 쓰는 것이 좋습니다.

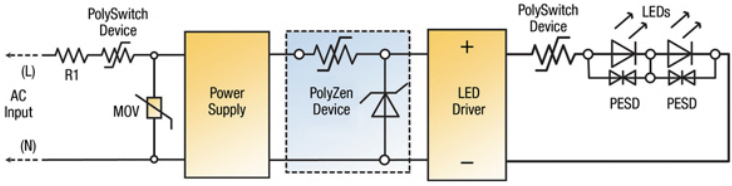
가격은 TVS 다이오드가 바리스터보다는 비싸므로, 신호를 보아가면서 적절히 선택해서 사용할 필요가 있습니다.

폴리스위치

퓨즈 대신 사용할 수 있는 전자식 퓨즈입니다. 일반 퓨즈는 한번 단선되면, 교체해주어야 하는 불편함이 있죠?

폴리스위치는 과전류가 유입되면, 열에 의해서 끊어집니다. 이후 과전류 원인이 사라져서, 열이 식으면, 다시 붙습니다. 즉, 교체해주어야 하는 불편함이 없는 아주 편리한 소자입니다. 이렇게 생겼습니다. (SMD 타입도 있어요.)

회로에서는 이렇게 사용합니다. (약간의 저항이 있습니다. 0 옴은 아니니까 참고하세요.)



전원 입력단이나, 과전류로부터 보호를 하고 싶은 소자의 전원단자에 직렬로 연결만 하면 됩니다. 알아서 끊어졌다 붙었다 하니, 완전 짱입니다. 1A 까지 견디는 것도 있고 2A 까지 견디는 것도 있으니까, 적절한 놈(?)을 선택해야 합니다. 너무 큰 용량을 선택하면 그냥 선으로 연결한 것과 동일하니까 주의하시구요.

스파크 킬러

코일을 쓰는 부품 (릴레이, 모터, 코일 그 자체)이 동작할 때, 인덕턴스 때문에 큰 에너지의 펄스 역전압이 발생합니다. 인덕턴스에서 역기전력에 의해 예상치 못한 큰 전압이 순간적으로 발생하여, 불꽃과 함께 글로우, ARC 등이 생기고, 고주파 진동이 일어나서 전자회로의 정상적인 동작을 방해하게 되는데, 이것을 막아주는 소자가 바로 스파크 킬러입니다.

쉽게 말해서, 불꽃발생을 방지하기 위해 접점간에 삽입하는 불꽃 소거 부품입니다.

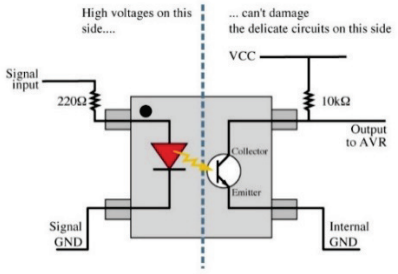
매우 다양한 형태가 있으며, 전기전자 상가에서 쉽게 구할 수 있습니다.

접촉기(마그네틱)에 붙이는 전용 스파크 킬러도 있습니다.

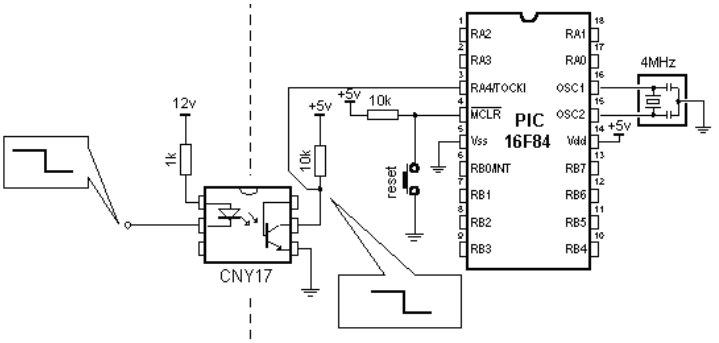
옵토커플러

다이오드와 TR 이 동시에 들어가 있는 부품이 있습니다. 옵토커플러 (또는 포토커플러 라고도 부름)인데요. 노이즈는 차단하고 신호만 전달하는 목적으로 자동화 분야에서 정말 많이 쓰이는 소자입니다.

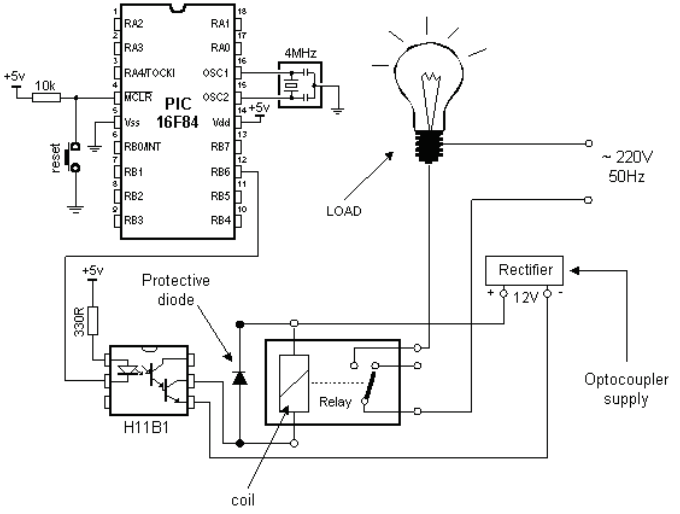
다른 말로 하면 "광학 커플" 이 되는데요. 잘 어울리는 두 소자가 커플이 되어서 한집에 살고 있는 것이죠.



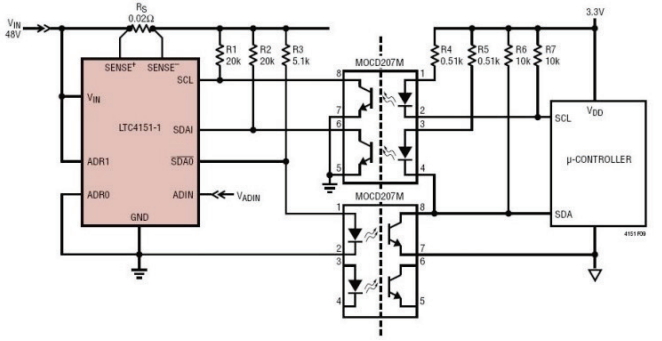
이 소자는 빛으로 전기신호를 전달할 수 있기에, 노이즈 대책회로에 중요한 역할을 담당하는 소자입니다. 내부에 있는 LED 를 켤 수 있을 정도의 전압과 전류만 인가한다면, 내부에 포토 TR 에 빛이 도달해서 TR 이 턴온 되는 구조입니다.



다음 그림처럼 출력회로에서도 옵토커플러를 이용할 수 있습니다.

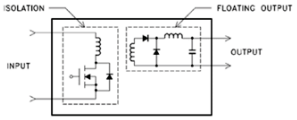


옵토커플러를 이용하면 아래 회로처럼 I2C 통신도 분리 (Isolated)시킬 수 있습니다. 아래 회로는 A/D 입력을 옵토커플러를 이용해서 전원분리한 예입니다.



DC/DC컨버터

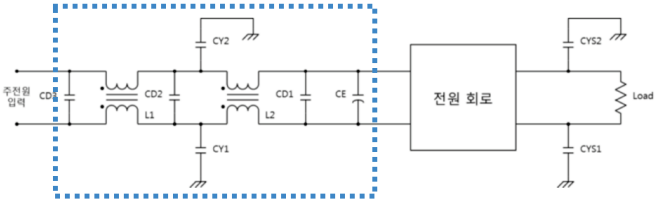
높은 전압에서 낮은 전압 (예: DC24V --> DC5V) 으로 낮출 필요가 있을 때 사용하는 부품입니다. 7805 등을 사용해서 간편하게 낮추는 방법도 있지만, 입력측과 출력측 그라운드가 분리되지 않기 때문에, 아이솔레이션 (노이즈 대책중 하나) 회로에서는 사용할 수 없습니다.



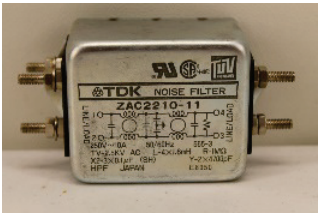
DC/DC 컨버터는 입력 전원과 출력 측의 그라운드가 서로 분리되어 있습니다.

노이즈 필터

전문가들은 전원쪽으로 들어오는 노이즈(EMI)를 감쇄시키기 위해 아래처럼 회로를 구성하도록 권장하고 있습니다.



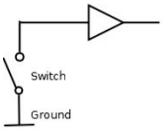
상당히 복잡해 보이죠? 그런데 이 고민을 한방에 해결해주는 부품이 있습니다. 바로 노이즈 필터죠. 앞 부분에 있어야할 회로가 이 안에 몽땅 다 들어있네요. 그냥 갖다 붙이면 됩니다.



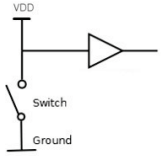
제 2 장. 초보전기전자

풀업 저항은 왜 필요한가?

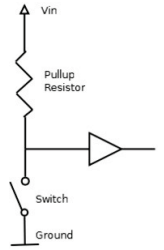
알고나면 별거 아닌데, 모를 때는 엄청 궁금했던 거중에 하나! 바로 풀업저항은 왜 필요한가? 입니다. 스위치를 누르면 Low 가되고 누르지 않으면 High 상태가 되는 회로를 생각해 보겠습니다. 일견 보기에는, 이렇게 회로를 꾸미면 될 것 같습니다.



그런데, 스위치가 눌려지지 않은 상태에서는 입력포트가 오픈(공중에 붕뜬)상태가 되고 맙니다. High 로 확실하게 해두지 않으면 Low 가 들어올 수 있습니다. 그래서 이렇게 회로를 꾸미면 해결될까요? 그러면, 스위치가 On 되었을 때, 전원간 쇼트가 발생하고 맙니다.



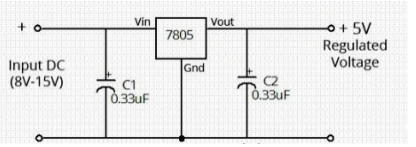
해결책은 바로 풀업저항입니다. 풀업저항을 사용하면 쇼트도 발생하지 않고, 스위치 OFF 상태에서 항상 High 를 공급해줍니다. 노이즈 대책 측면에서도 좋습니다. 풀업저항으로 흐르는 전류보다 약한 전류의 노이즈는 들어올 수 없게 해줍니다.



풀업저항은 4.7K 오옴~10K 오옴 사이가 적당합니다.

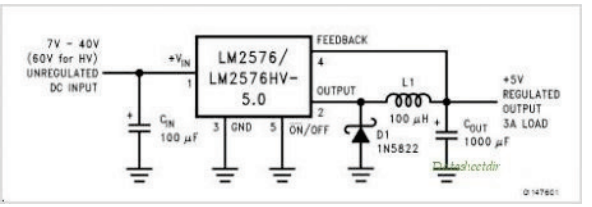
5V 만들기

9V 이상 전압 입력을 받아서 5V 를 만드는 방법을 알아보겠습니다. 우선, 7805 레귤레이터를 이용하는 방법이 있습니다.



이 방법은 열이 발생하므로, 방열판을 부착해야 됩니다. 전압도 15V 까지만 사용할 수 있습니다. 방열판은 공간을 많이 차지하고, 발생된 열이 많을 경우 팬도 설치해야 합니다.

LM2576 이라는 소자도 5V 를 만드는데 자주 쓰입니다. 이 소자는 스위칭 레귤레이터라서 주변에 부품이 좀 많이 들어갑니다. 대신, 거의 열이 발생하지 않고, 효율도 좋은 편입니다.



위 2 가지 방법은 입력측과 출력측 그라운드가 서로 연결되어 있어서, 아이솔레이션 회로에서는 사용할 수 없습니다. 아이솔레이션 회로에서는 isolated DC/DC 컨버터를 사용해서 5V 를 만드는 방법을 씁니다.

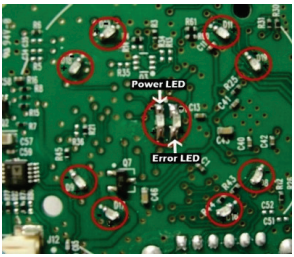
디버깅 테크닉

MCU 또는 큐블록 코어는 자신의 에러를 표출할 방법이 없습니다. LCD 라도 하나 달려있으면 좋으련만..

개발당시에는 여러가지 디버깅 방법이 있지만, 필드에 설치된 후에는 마땅히 내부 동작상황을 살펴볼 방법이 없습니다. "이 분야의 고수"라면, 필드에서 실행 중 보드의 동작상황을 파악할 수 있는 장치(?)를 마련해 둘 것입니다.

가장 간단하게 사용할 수 있는 방법은 LED 하나 또는 2 개를 연결해 두는 것입니다. LED 2 개만으로도 4 가지의 상태를 표현할 수 있습니다.

- OFF OFF : 정상동작
- ON OFF : 상태 1
- OFF ON : 상태 2
- ON ON : 에러



여기에 깜빡임까지 표현해준다면, LED 두개로 상당히 많은 정보를 표현할 수 있습니다. 이 LED 를 구지 최종유저가 볼 수 있도록 해줄 필요는 없구요. 그냥 제품케이스를 열어서 여러분만 볼 수 있는 위치에 달아두십시오.

두번째 방법으로는 사운드를 이용하는 방법이 있습니다.

이 방법은 포트 하나면 됩니다. "부저"라는 소자가 있습니다. HIGH 만 공급해주면 "삑~"하고 소리가 들립니다. (PIEZO 말고 BUZZER 입니다. PIEZO 는 펄스를 내보내줘야 소리가 납니다. 반면 BUZZER 는 내부에 발진 회로가 들어있어서 HIGH 만 공급해주면 소리가 납니다.)

평상시는 소리가 나면 안되겠죠. 에러발생시 "삑~"하는 소리가 들리도록 하는 겁니다. 소리의 주기를 조정한다면 다양한 정보를 표출할 수 있습니다. "삑, 삑, 삑, 삑" 네번 들린다면 "에러상황 2 번"이라고 나름대로 정의하시면 됩니다.

세번째 방법은 LCD 를 연결할 수 있는 포트를 만들어 두는 것입니다. 일반적인 캐릭터 LCD 는 최소 7 가닥의 연결선이 필요하고, LCD 관련 초기화 루틴, 구동 루틴을 다 작성해주어야 합니다.

더 쉬운 방법이 있습니다. 저희 회사에서 판매하고 있는 시리얼캐릭터 LCD 인 CLCD 제품은 I2C 시그널만 출력해준다면, LCD 에 문자/숫자를 자유롭게 표현할 수 있습니다.

4 가닥으로 메인 PCB 와 연결합니다. 5V, GND, SCL, SDA 단자입니다. 5V 와 GND 로 전원을 공급하고, SCL, SDA 로 I2C 파형을 보내면 LCD 화면에 문자 및 숫자가 표시됩니다. 저희 제품 큐블록의 경우엔 CLCD 를 간단하게 제어할 수 있는 전용 명령어를 갖추고 있습니다.

제품에는 CLCD 를 연결할 수 있는 포트만 있는 채로 출하하시고, 상태 체크가 필요할 때 CLCD 모듈을 가지고 가서 연결한 뒤, 내부상태를 보는 식으로 활용하세요.

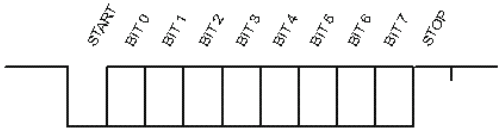
RS232C란 무엇인가?

MCU 응용회로와 자동화 분야에서 광범위하게 사용하고 있는 RS232C 란 무엇인지 살펴보겠습니다.

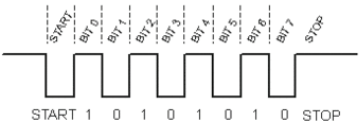
1. 직렬 통신 방법 중 하나입니다. 병렬통신은 여러가닥으로 신호를 전달하는데, 대표적인 병렬통신은 예전에 쓰였던 프린터 포트, IDE 포트 (하드디스크 연결용) 이 있었습니다만 지금은 안씁니다.

2. 직렬통신은 말 그대로 직렬 (Serial)로 데이터를 송수신 하기 때문에 신호선이 2,4 가닥으로 해결됩니다. 우리가 아는 대부분의 통신방식은 다 직렬통신입니다. 이더넷, USB, SATA, SPI, I2C 등

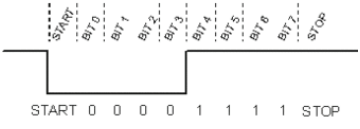
3. RS232C 는 가장 오래된 직렬 통신 방법입니다. 단 2 가닥 (TX, RX)로 데이터를 보내는데, 신호가 +/- 12V 입니다. GND 선까지 최소 3 가닥이 있어야 통신이 됩니다.



4. 위 그림처럼 Start 신호를 시작으로 0 비트부터 7 비트까지 차례대로 나갑니다. 맨뒤에 Stop 비트가 따라옵니다. 아래 그림은 실제로 데이터가 나가는 모양을 표현한 것입니다.



16진: 55 2진 : 01010101

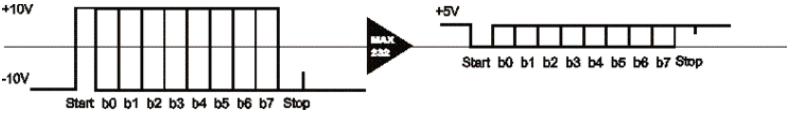


16진: F0 2진 : 11110000

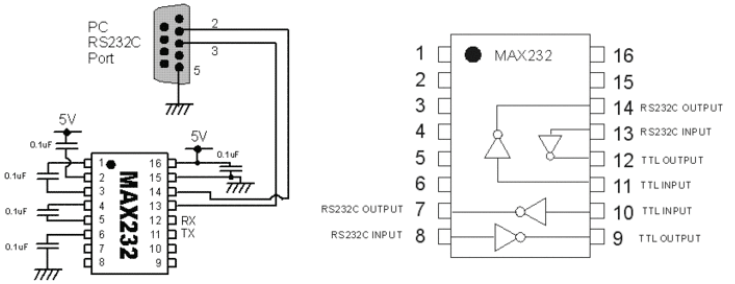
5.보내고 받는 측 사이에 약속이 있어야 합니다. 속도, 비트수, 에러 체크방법(패리티) 등이 있는데, 쌍방간에 알고 있어야 통신이 됩니다.

6. MCU 등에서는 USART 라고 부릅니다. USART 는 universal synchronous /asynchronous receiver transmitter (범용 동기/비동기 송수신기) 를 뜻합니다. MCU 는 5V 레벨로 신호를 주고 받기 때문에, RS232C 선로에 바로 물릴 수 없고 별도의 전압변환칩을 필요로 합니다.

7. 대표적인 RS232C 용 전압변환칩은 MAX232 칩이 있습니다. MAX232 칩을 쓰면 5V 신호를 RS232C 신호 레벨인 +/-10V 로 바꿔줍니다.



MAX232 연결회로



8. RS232C 는 1:1 통신 방법입니다. 1:N 통신이 안됩니다.

9. 거리는 스펙상으로 대략 15 미터 까지 허용됩니다. 실제로는 6~7 미터 미만에서만 사용을 권장하고 있습니다. 그보다 긴 거리가 필요하면 노이즈에 강한 RS422 이나 RS485 로 갈아 타야됩니다.

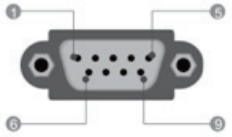
10. RS232C 가 아직도 많이 쓰이는 이유? 그건 아주 간단한 통신 방식이기 때문입니다. 지금도 많은 산업용 기기들은 RS232C 통신으로 서로 데이터를 교환합니다.

11. 최대 통신속도는 230,400BPS 까지 되는데, 보통은 이렇게 까지 속도를 높이지 않고, 115200 이나 57600 정도를 많이 씁니다. 이유는 전압 전송 방식이여서 노이즈가 많이 실리기 때문입니다.

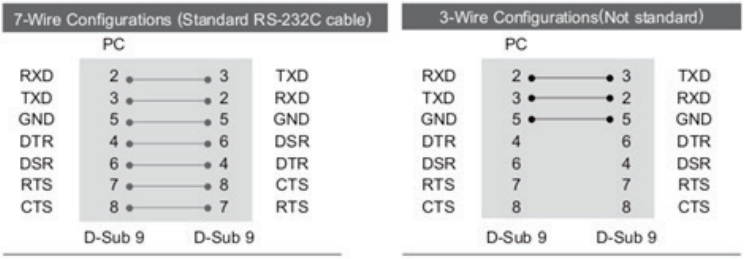
12. PC 에 COM 포트를 보면 3 핀 (RX, TX, GND)가 아니라 9 핀입니다. 나머지 핀들은 상대측의 상태를 알기위한 보조핀입니다. RTS (Ready to send) 는 보낼 준비가 됐는지 체크하는 역할입니다. 그런데 이러한 보조핀들을 잘 안쓰는 경우도 많습니다. 그냥 3 가닥만 가지고도 충분히 통신이 가능합니다.

아래 그림은 DTE (주로 PC 쪽)의 D-SUB 9 핀 RS232C 핀배치도입니다.

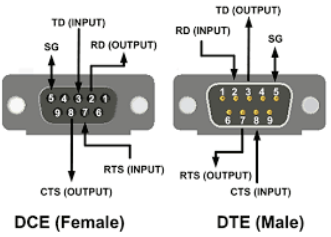
No.	Pin Name
1	No connection
2	RXD (Receive data)
3	TXD (Transmit data)
4	DTR (DTE side ready)
5	GND
6	DSR (DCE side ready)
7	RTS (Ready to send)
8	CTS (Clear to send)
9	No Connection



RS-232C Configurations

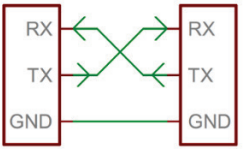


13. DCE 는 Data Communications Equipment 의 약자인데 주로 모뎀(장비)측을 의미하고, DTE 는 Data Teminal Equipment 의 약자인데 주로 PC 측을 의미합니다. 그래서 (아래 사진을 참조) PC 쪽 RS232C 코넥터는 Male 로 되어 있습니다.



안정적인 통신 프로그램 작성법

두개의 장비가 있고 서로 데이터를 교환할 필요가 있을 때, 가장 싸고 쉽게 해결할 수 있는 방법이 RS232C 를 사용해서 두 장비를 서로 연결하는 방법입니다.



위 그림처럼 단 3 가닥만 있으면 됩니다. 그럼 하드웨어적으로 결선이 끝났으니 이제 프로그램을 작성하는 일만 남았죠. 어떻게 데이터를 주고받아야 안정적으로 문제없이 계속 데이터 교환이 가능할까요? 생각보다 간단하진 않습니다.

안정적인 통신 프로그램을 작성하는 방법을 하나씩 알아보겠습니다.

1. 체크섬 보내기

데이터 전송중에 노이즈에 의해서 파손된 데이터가 전달되는 상황을 감안하지 않을 수 없습니다. 특히 통신선로는 외부에 노출이 되어 있어서, 노이즈로부터 공격 받기 쉽습니다. 그래서 모든 데이터 뒤에는 각 데이터의 합 (Check SUM)을 보내도록 하는게 좋습니다. 받는 측에서는 이를 검사해서 데이터의 신빙성 여부를 판단하면 됩니다. 체크섬이 안맞으면 이번에 도착한 데이터는 신빙성이 없으므로 다 버리도록 하면 됩니다.

2. 타임아웃 설정하기

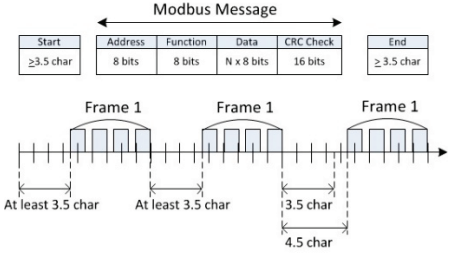
받는 쪽에서 생각해보겠습니다. 데이터 10 바이트가 와야되는 상황이라고 했을때, 평상시에는 10 바이트씩 꼬박꼬박 잘오다가, 갑자기 5 바이트만 오고 그 다음 데이터가 안들어오면 어떻게 하실건가요? 갑자기 선이 단선될 수도 있고, 보내는 장치의 전원이 꺼질 수도 있고, 여러가지 다양한 원인에 의해, 정해진 바이트수가 다 안들어 오는 상황은 많이 생길 수 있는 일입니다.

그래서 타임아웃이 필요합니다. 받는 쪽에서는 무작정 다음 바이트가 올때까지 무한루프로 기다리는 식으로 프로그램을 작성해선 안됩니다. 일정시간을 정해놓고 (예를 들면 1 초) 그 시간만 기다린 뒤, 그때까지도 데이터가 안오면 통신실패 (Fail) 로 간주하는 식으로 프로그램을 작성해야 합니다.

타임아웃을 고려하지 않는다면, 무한루프로 빠져서 멍청하게 대기만 타는 장비를 만들 가능성이 있습니다.

3. 프로토콜

교환할 데이터가 조금 복잡한 경우에는 프로토콜까지 만들어야 하는 상황이 발생할 수 있습니다.



어드레스, 펑션, 데이터, 체크섬까지 하나의 패킷으로 묶고 패킷과 패킷사이는 간격을 떨어뜨려서 패킷을 구분하도록 하는등, 이런 일련의 약속을 정해서 보내는 곳과 받는 곳에서 그 약속을 지키는 것을 프로토콜이라고 합니다.

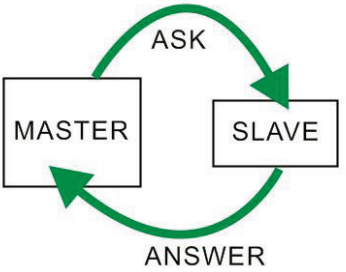
프로토콜을 만들기 위해서는 나름 고려할 사항도 많고, 생각할 부분이 많기 때문에, 여러분이 새로운 프로토콜을 만들 필요없이 이미 나와있는 프로토콜중 하나를 쓰시면 편합니다.

저는 MODBUS 를 추천합니다. MODBUS 도 종류가 많은데 MODBUS RTU 가 가장 간결하고 짧게 데이터를 주고 받습니다. MODBUS ASCII 를 쓰면 쓸데없이 데이터가 길어질 수 있습니다. MODBUS RTU 를 사용하면 CRC 라는 검증방법이 들어가 있기 때문에, 별도로 체크섬을 사용하실 필요가 없습니다.

모드버스란 무엇인가?

통신을 원하는 두개의 디바이스가 있습니다. 둘 중 어떤 놈이 먼저 말을 걸까요? 먼저 말을 거는 놈을 "마스터 (또는 클라이언트)"라고 칭하겠습니다.

듣는 쪽에서는 대답을 해주어야 겠지요. 이놈을 "슬레이브 (또는 서버)"라고 부르겠습니다.



그럼 뭐라고 말을 걸까요? 마구잡이로 숫자를 보내면 받는측에서 갸우뚱할 겁니다.
서로 간의 약속이 필요합니다. 이것을 프로토콜이라고 하죠.

편하게, 첫번째 바이트는 "주소", 두번째 바이트는 "코멘드", 세번째 바이트는 "데이터라고 정해보겠습니다.

이렇게 약속을 정했으니, 받는 측에서도 3 바이트가 오면, 그에 상응하는 동작을 하면 됩니다. 아주 옛날에는 업체마다 이러한 프로토콜이 제각각 이였습니다.

그러다 어떤 업체가 이러지 말고 표준을 정하자고 나섰습니다. 그 회사가 "모디콘"입니다. 그리고 이름을 "모드버스 (MODBUS)" 라고 붙입니다. 그리고 프로토콜을 자기 맘대로 만들어버립니다.

START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 CHAR :	2 CHARS	2 CHARS	*n* CHARS	2 CHARS	2 CHARS CRLF

처음에 사람들은 콧방귀를 뀝니다. "흥~ 지가 뭔데, 자기를 따라하라 마라 그러지?"

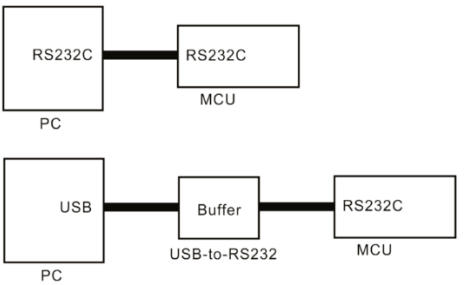
근데, 스팩을 보니까 그럴싸합니다. 하나둘씩 따라하는 업체가 생겨납니다. 세월이 수십년이 지났습니다. 지금은 대다수의 업체에서 이 프로토콜을 따라하고 있습니다. 자발적 "산업표준"이 생겨난 겁니다.

이게 바로 모드버스입니다.

USB to RS232C 케이블 사용시 주의할 점

요즘 노트북에는 RS232 포트가 없습니다. 데스크탑에도 없는 경우가 있습니다. 그래서 USB 포트에 꼽아 RS232 를 만들어주는 USB-to-RS232 케이블을 많이 사용합니다.

일반 유저들은 드라이버만 제대로 설치하면, 실제 RS232C 포트처럼 편리하게 사용할 수 있습니다. 만약 MCU (또는 큐블록)보드에서 이 케이블을 통해서 PC 와 통신하려고 한다면, 몇 가지 고려해야될 사항이 있습니다.

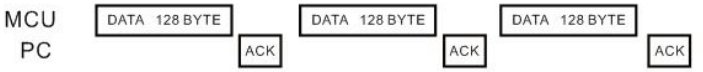


위 그림처럼 USB-to-RS232C 케이블 안에는 별도의 버퍼가 있습니다. MCU 에서 보낸 데이터는 케이블안에 있는 버퍼에서 버퍼링을 합니다. 그리고 PC 가 받을 준비가 되었을 때 보내줍니다.

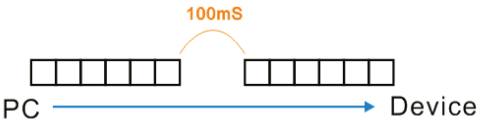
버퍼에는 한계가 있습니다. 통신속도가 너무 빨라 버퍼에 차는 속도를 USB 에서 읽어가는 속도가 따라가지 못할 때, 버퍼 오버플로우가 발생되고 그 이후에 수신되는 데이터는 버려지고 맙니다.

한꺼번에 많은 양의 데이터를 쉬는 구간없이 보내게 되면, USB-to-RS232C 안에 있는 버퍼가 금방 꽉차게 되서 데이터를 다 못보내는 수가 있습니다.

그래서 MCU 에서 PC 로 데이터를 보낼 때는 중간중간 끊어서 PC 에서 잘 수신하고 있는지 ACK 를 받는 방식으로 송신을 해야 안전합니다. PC 에서는 ACK 를 보내기 위해서 쉬는 시간이 생기게 되고, 이 타이밍에 USB-to-RS232C 버퍼에 있는 모든 데이터를 읽어오게 됩니다. 한마디로 숨쉴 틈을 주고 데이터를 보내야 한다는 뜻입니다.



두번째로 고려할 사항은 인터벌에 관한 것입니다. 저도 이것 때문에 고생한적이 몇번 있는데요. PC 와 디바이스가 RS232C 로 직접 연결되었을 경우, 전송데이터 사이의 딜레이는 그대로 유지됩니다.

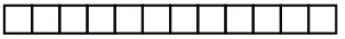


앞 그림은 PC 에서 6 바이트의 데이트를 보내고, 100mS 간격을 둔 뒤, 다시 6 바이트를 보내는 상황입니다.

RS232 통신에서는 프레임 (의미 있는 데이터의 집합) 구분을 위해 이런식으로 인터벌을 두고 데이터를 보내는 상황이 많습니다. MODBUS-RTU 도 이러한 방식의 프로토콜입니다.

그런데, USB-to-RS232 케이블을 쓰면, 이 인터벌간격이 틀어지는 경우가 생길 수 있습니다. 이유는 USB-to-RS232C 안에 있는 MCU 가 PC 와 통신을 하기위해서 중간중간 나름의(?) 작업을 하느라고 그렇습니다.

그래서 위와 같은 경우, 아래 그림처럼 전혀 엉뚱한 인터벌을 포함한 형태로 전송하기도 하고, 아예 붙여서 보내지기도 합니다.



따라서, USB-to-RS232 케이블을 사용한다면, 인터벌을 가지고 프레임을 구분하는 것보다, 데이터의 내용물을 가지고 프레임을 구분하도록 코딩을 하셔야, 에러가 발생하지 않습니다.

그래서 저는 CUBLOC 의 MODBUS-RTU 수신 분석을 할 때, 인터벌을 조사하지 않고, 각 프레임 뒤에 붙어오는 CRC16 을 가지고, 한 프레임의 끝이 어딘지 판단하도록 코딩을 하였습니다.

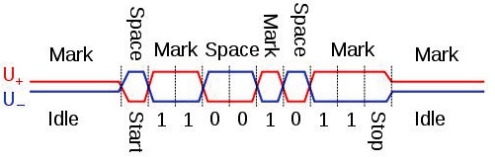
노이즈에 강한 통신 RS485

RS232C 통신은 최대 15 미터까지 사용 가능하다고 스팩에 나와있지만, 제가 실험해본 바로는 안정적으로 사용하려면 6, 7 미터 이내에서만 사용해야 했습니다. 그 이상의 거리에서는 노이즈가 많이 타서, 보레이트를 늦추거나, 패리티를 설정해야 하는 등의 추가적인 대책이 필요했습니다.

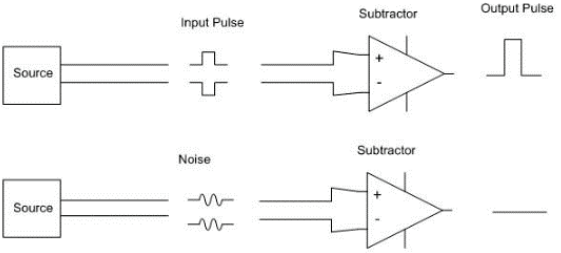
그보다 더 먼 거리를 서로 연결하기 위해서는 노이즈에 잘 견디는 다른 방법이 필요합니다. 그래서 나온 것이 RS422 과 RS485 가 있습니다.

- RS422 : 전이중 통신방식, 최대 1.2Km, +/- 7V
- RS485 : 반이중 통신방식, 최대 1.2Km, +/- 5V

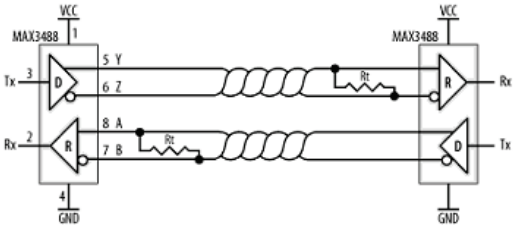
두개의 선을 이용해서 하나의 시그널을 만드는데, 한선에는 원래신호, 다른 한선에는 반전신호를 실어주는 방식입니다. (이더넷, USB 모두 이 방식을 사용합니다.)



이렇게 되면, 노이즈가 실려도 같이 두 선에 동일하게 실리기 때문에, 논리적인 변동은 없게 됩니다. 그래서 노이즈에 강한 신호 전달방식이 되겠습니다.



MCU (큐블록)에서는 5V 레벨의 송수신 파형을 취급하는데, 이를 USART 또는 편의상 5V RS232C 라고도 부릅니다. 여기에 RS232C 신호로 레벨변환을 해주는 칩 (예:MAX232)을 붙이면 RS232C 가 되는 것이고, 여기에 RS422 신호로 레벨변환을 해주는 칩 (예:MAX3488)를 붙이면 RS422 이 됩니다.

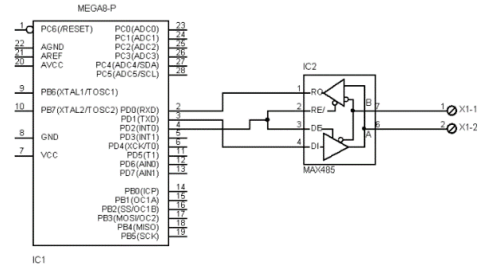


RS422 은 TX 선과 RX 선에 각각 2 가닥씩 할당됩니다. 따라서 보내고 받고가 동시에 되는 전이중 통신이 가능합니다. 이점에 있어서는 RS232C 와 동일합니다.

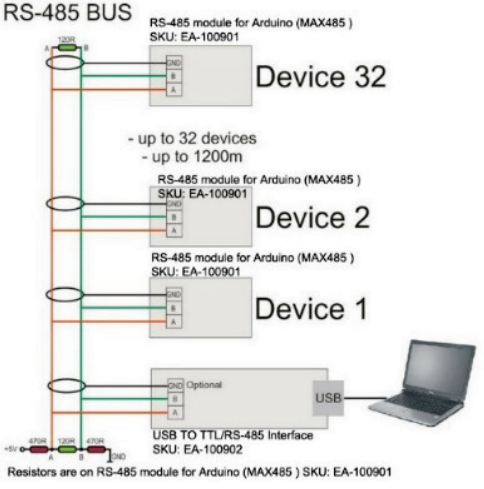
기존 RS232C 로 되어 있는 선로를 거리만 늘리고 싶다고 했을때는, RS422 로 바꾸는 것이 좋습니다. RS232-TO-RS422 컨버터만 붙이면 신경쓸 일이 없게 됩니다.

반면 RS485 은 단 2 가닥으로 송신과 수신을 모두 하는 방식입니다.

그래서 송신과 수신을 동시에 할 수가 없고 TX 와 RX 중 어떤게 선을 점유하느냐의 문제가 생깁니다. 그걸 결정해주는 핀이 별도로 있습니다. (RE/ 와 DE 핀을 보통 묶어서 사용합니다.)



이러한 불편함에도 불구하고 RS485 가 많이 쓰이는 이유는 단 2 가닥으로 1:N 통신을 쉽게 할 수 있다는 장점 때문입니다.



그리고 통신선로 양쪽 끝에 종단저항 (120 오옴 정도)은 반드시 붙이세요. 종단 저항은 신호의 메아리현상을 막아주고, 노이즈가 잘 침투하지 못하도록 해줍니다.

제 3 장. 노이즈

노이즈란 무엇인가?

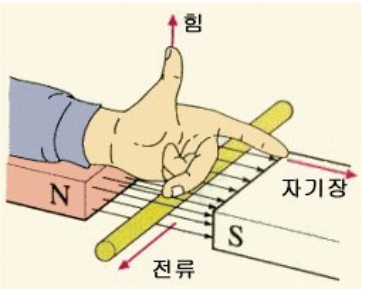
전기 전자 분야 엔지니어를 끝까지 괴롭히는 노이즈 (아주 지긋지긋 하죠!) 성가시다고 외면하지 마시고 마치 친구처럼 친근하게 생각해보세요. 노이즈에 대해서 알면 알수록 노이즈 대책을 잘 세울 수 있게 되고, 출하 후 A/S 횟수를 줄이는데 도움을 줍니다. 이것은 곧 수익과도 직결되는 사안입니다.

실력있는 엔지니어란 "노이즈에 대한 풍부한 경험과 처리 능력"을 보유하여, 완성도가 높은 제품, 고장이 잘 안나는 제품을 설계할 수 있는 사람을 뜻합니다. 노이즈를 정복하기 위해서는 노이즈의 실체부터 알아봐야 겠지요.

노이즈란?

시스템을 방해하는 모든 전기 전자적인 요인을 뜻합니다. 전기 전자기기는 단순히 전기/전자가 흐르는 장치가 아닙니다. 전류가 흐르면서 주위에 전계와 자기가 유도됩니다.

그 유명한 플레밍의 왼손법칙을 아실겁니다. 전기가 흐르면 필연적으로 자기도 발생합니다. 또한 전선 주위에 자기가 있으면 없던 전류도 흐릅니다. 이런 복잡한 상황이 여러분이 설계한 제품안에서 항상 벌어지고 있습니다.



즉, 개발자가 의도하는 바와 상관없이 전류가 흐르는 도선주위에 발생하는 불필요한 에너지가 전자파 노이즈의 근원이 됩니다.

이렇게 발생된 노이즈는 전달 경로를 통해 다른 기기에 전달되면서 성능저하나 오동작의 원인이 됩니다.

이것을 "전자파 장애" 또는 EMI (Electro Magnetic interference)라고도 부릅니다.

이러한 전자파 장애의 피해를 줄여보고자 정부기관에서 만든 것이 FCC, KCC, CE 와 같은 EMI 인증입니다. 우리나라에서 만든 대부분의 전자제품은 KCC 인증을 통해, 외부로 불필요한 전자파 노이즈를 발생하지 않는 여부를 심사하여 인증을 내어 줍니다.

수입하는 제품도 마찬가지입니다. 아이폰 수입할 때 전자파 인증을 거치는 것을 아실 겁니다. 미국에서 FCC를 득했다고 하더라도, 우리나라에는 KCC라는 기준이 있고, 모든 수입전자기기는 KCC 인증을 득해야 판매가 가능합니다.

정부차원에서 이런 노력을 하는 이유는 대 혼란을 막기 위해서입니다.

요즘은 가전/자동차/산업 등의 모든 분야에서 전자기기를 쓰지 않을 수가 없는데, 이런 최소한의 전자파 장애 제한노력이 없었다면, 지금쯤 엄청난 전자파 공해속에서 살고 있었을 것입니다.

이런 노력에도 불구하고, 노이즈는 매우 다양한 발생요인을 가지고 있고, 다양한 경로로 침입하기도 하며, 기기 내부의 다른 곳에서 발생하는 노이즈의 영향도 받기 때문에, 노이즈 대책은 필수적으로 필요한 것입니다.

노이즈의 종류

노이즈는 "자연 노이즈"와 "인공 노이즈"로 크게 분류합니다. "자연노이즈"는 말그대로 자연적으로 발생한 노이즈를 뜻하는데, 번개 (낙뢰)등이 대표적인 예입니다.

"인공노이즈"는 사람들이 만들어낸 전자기기 등에서 발생하는 "전자파 노이즈" 입니다.

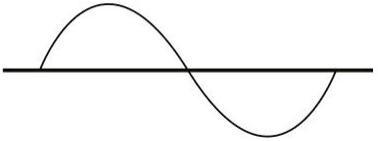
인공 노이즈는 "방사 노이즈"와 "전도 노이즈"가 있습니다. "방사 노이즈"는 방송이나 휴대무선기등의 통신전파에 의해 발생하기도 하며, 송전선의 코로나 방전, 형광등의 안정기에서 발생하는 노이즈, 오토바이의 점화시에 발생하는 노이즈등, 공간으로 방출되는 노이즈를 뜻합니다.

"전도 노이즈"는 전자제품안에 전선이나 통신선, 전원선 등을 통해 유도되는 전자파를 뜻합니다. 전도 노이즈는 Transient, Impulse 등과 같이 전기선 양단에 흐르는 Normal Mode Noise (노멀 모드 노이즈)와 전기선을 통해 Line 과 어스간에 전달되는 Common Mode Noise (커먼 모드 노이즈)로 구분됩니다.

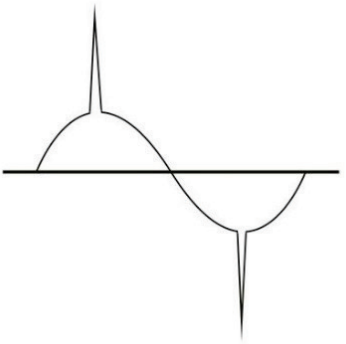
노멀 모드 노이즈는 특히 전송중에 있는 데이터에 치명적일 수 있습니다. 우리가 아는 대부분의 통신선이 노멀모드 노이즈의 공격대상이 됩니다. RS485 라인, 이더넷, USB 등 두 가닥사이의 전압차이로 신호유무를 판단하는 통신선로들입니다.

커먼 모드 노이즈는 Ground 를 기준전위로 채택하는 MCU, Memory, Logic 회로등이 내장된 전자기기에 에러를 유발시킬 수 있습니다. 즉, 거의 모든 전자기기들이 커먼 모드 노이즈의 공격대상이 됩니다.

노이즈의 유형

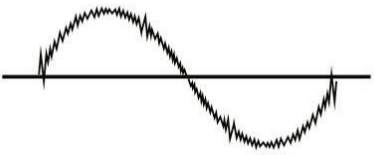


원래의 신호파형이 아래와 같을 경우 크게 3 가지 경우의 노이즈가 발생될 수 있습니다.



1. **IMPULSE 의 경우** : 갑자기 큰 전압이 원래 파형에 섞여서 오는 경우입니다. 모터용 인버터가 기동하는 순간, 또는 SWITCH 가 ON/OFF 되는 순간, 아니면 코일이 있는 전자부품이 ON/OFF 하는 순간에 많이 발생됩니다.

이러한 IMPULSE 는 MCU 와 같은 섬세한동작을 요구하는 부품에 치명적인 영향을 주어, RESET 이 된다던가, 순간적으로 멀펑션상태에 빠지는등의 이상 동작을 유발시킵니다. 심지어 FLASH 메모리로 된 MCU 의 프로그램메모리에 손상을 주어, (다시 프로그램을 라이팅하지 않는한) 영구적으로 동작불능의 상태로 빠지게도 만듭니다.

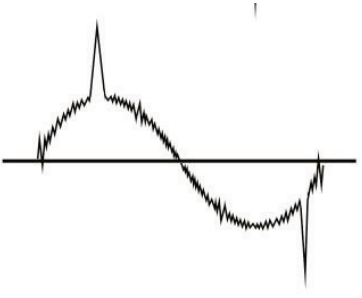


2. **전자파 (EMI)가 실려있는 경우** : 전반적으로 신호에 잡음이 끼어 있는 상태입니다. 전원선에 이런식으로 노이즈가 끼어있는 상태로 메인장치에 공급이 된다면, 동작이 불안하게 됩니다.

특히 A/D 변환을 해서 그 결과를 처리하는 장치라면, A/D 측정값을 흔들리게 하므로 제품 고유의 성능을 제대로 발휘할 수 없을 정도의 치명적인 영향을 주게 됩니다.

통신선로에 이러한 노이즈가 끼어있다면, 1 을 0 으로, 0 을 1 로 판단하는 일이 생기고 맙니다. 원래 값과 전혀 다른 엉뚱한 데이터를 송신하고, 수신하게 되므로, 정상적인 동작을 보증할 수 없게 됩니다.

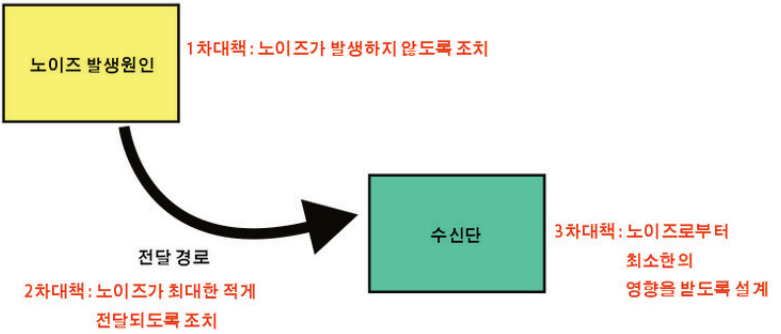
음향장비에도 이러한 노이즈는 치명적입니다. 깨끗한 음질의 사운드를 들을 수 없게 됩니다.



3. **노이즈와 임펄스가 같이 오는 경우**: 가장 최악의 상황입니다. 이런 상황에서 아무런 노이즈 대책이 없다면, 해당 전자기기의 정상적인 동작을 보증할 수 없습니다.

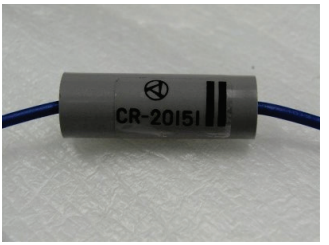
노이즈 대책 개요

3 단계에 걸쳐 노이즈 대책을 세웁니다.



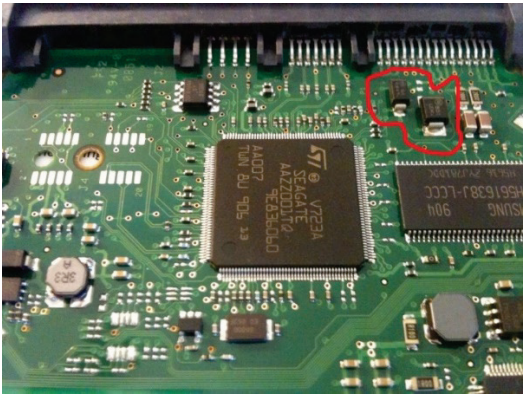
1 차로 노이즈는 발생단계부터 원인을 제거하려는 노력을 하는 것이 좋습니다.

"스파크 킬러"가 좋은 예입니다. 릴레이, 접촉기 (마그네틱)등은 ON /OFF 시에 스파크가 발생합니다. 이 스파크는 방사노이즈, 전도 노이즈의 형태로 다른 기기에 영향을 주기 때문에, 발생이 안되도록 하는게 좋습니다. 스파크 킬러라는 간단하면서 싼 부품으로 이것을 해결할 수 있습니다.



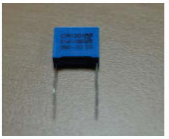
2 차로 필터기술 (Filterling) 를 통해 노이즈가 통신선로/전원선로 등을 통해서 수신단으로 전달되는 것을 최소화 시켜줍니다. 케이블 양쪽 끝에 있는 페라이트 코어가 좋은 예입니다.

3 차로 수신단에 도착한 노이즈를 내부로 들어오지 못하도록 차단하는 차폐기술(SHIELDING)입니다. 앞서 소개 드렸던 TVS 다이오드등이 좋은 예입니다.



스파크 킬러 사용법

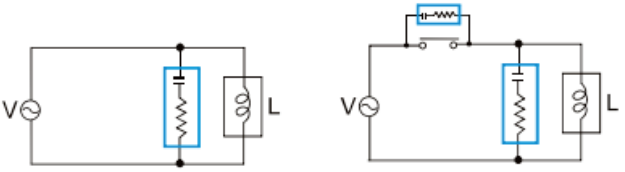
릴레이를 외부에 연결하는 경우, 스파크 킬러는 필수적으로 부착 해주어야 합니다.



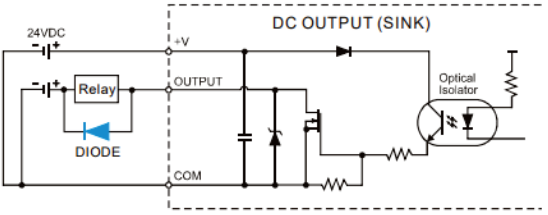
릴레이가 On/Off 할때 발생되는 스파크를 제거해주는 역할을 합니다. 이는 릴레이 수명연장에 도움을 줄 뿐만 아니라, 이 스파크가 메인칩까지 흘러들어서 칩이 리셋하거나 오동작하는 현상을 막아주는 중요한 역할을 합니다.

간혹, 스파크 킬러를 붙여주지 않고, 저희 제품이 노이즈에 약하다고 말씀하시는 분들이 있는데, 이는 잘못된 생각입니다. 아무리 좋은 장비라도 스파크 킬러는 반드시 붙여주어야 합니다.

교류 회로에서 스파크 킬러 부착 위치입니다.

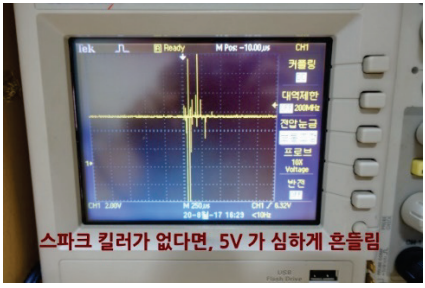


직류회로에서는 다이오드 하나로 스파크를 잡을 수 있습니다.



스파크 킬러가 없는 경우 어떤 상황이 벌어지는 한번 보도록 하겠습니다.

스파크 킬러가 없다면, 콘트롤러의 내부 5V 전원은 아래사진에서 볼 수 있듯이 상당히 심한 노이즈가 들어옵니다. 이로인해 콘트롤러가 오동작하기도 하고, 심한 경우 메인칩이 파손되는 경우도 있습니다.



전기상가에서 쉽게 구할 수 있는 스파크 킬러만 붙여준다면, 노이즈가 전혀 발생하지 않고, 다음 사진처럼 깨끗한 5V 파형이 유지됩니다.

제어반에서의 노이즈 대책

제어반, PLC, HMI 을 쓰는 시스템에서 노이즈 대책이 중요하죠. 여기 미국 "콘트롤 디자인" 메거진에 소개된 "어떻게 하면 노이즈를 줄일 수 있나 ? (How can we reduce the noise) 라는 제목의 기사를 한번 보겠습니다.


control design FOR MACHINE BUILDERS

LOGIN | REGISTER

Connections Control Design Motion Sensing Safety & Security Displays

Home / Articles / 2008 / How can we reduce the noise?

How can we reduce the noise?

Alternatives to help sort out the best course of action for resolving electrical noise interference from motors and drives that affect sensor signals and HMI

Nov 26, 2008

Print Email Tweet Share G+ Share

Electrical noise interference from motors and drives is affecting our sensor signals and HMI too often, the sensor signals and even the HMI of our metal-cutting machines go haywire, largely because of electrical noise interference from many old, large, nearby motors and drives that we really can't replace. One solution suggested is to start using fiberoptic cable for signal transmission, but that's not cheap. Alternatively, there also are dozens of power filters, uninterruptible power supplies and other power conditioning devices out there, ranging in price from less than $100 to more than $1,000. We could use some information about both alternatives to help sort out the best course of action.

White paper download: Wireless Connectivity for the Internet of Things

ANSWERS

Try These First

Following are a few basic design rules that might take care of the problems.

Separation between the low voltage control and high voltage power lines: If those lines need to cross, they should do so at 90° angles.

Use shielded cables and/or shielded cable trays. It is important to note that simply using what is typically sold as shielded sensor cable could actually increase the problem, especially when using cables with M12 connectors. Most such cables do not connect the shield to the coupling nut and, as a result, do not provide a clean path to ground.

Provide for solid machine ground interconnections. This is a must when using shielded cables or the shield will serve to equalize the potential between the different machine sections.

요점만 번역해보겠습니다.

모터 및 드라이브를 사용하는 시스템에서 모터 기동시, 센서신호및 PLC/HMI 동작에 영향을 주는 경우가 많습니다. HMI 와 PLC 가까이 설치된 모터및 드라이브의 전기 노이즈 간섭으로 때론 치명적인 영향을 주기도 합니다. 모든 선을 광섬유 케이블로 바꾸고, 수십개의 전원필터와 무정전 전원장치(UPS)를 사용한다면 노이즈를 상당부분 잡을 수 있지만, 이것은 100 달러에서 1000 달러 이상 소요되기에 현실적으로 어렵습니다.

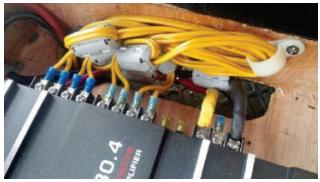
고 비용이 필요한 방법이 아닌, 차선책 몇 가지를 소개합니다.

제어반 안에서 선정리 하실때, 저전압(신호선)과 고전압(전원선, 동력선)은 분리해야합니다. 어쩔 수 없이 교차해야 한다면 90 도로 교차해야 합니다.

차폐(실드) 케이블 및 차폐 덕트를 사용하세요. 대부분의 케이블들은 실드가 안되어 있으므로 주변에서 발생하는 노이즈가 여과없이 신호에 실리고 맙니다. 차폐(실드)케이블의 실드부분은 접지를 해야 실드의 효과가 발생합니다.

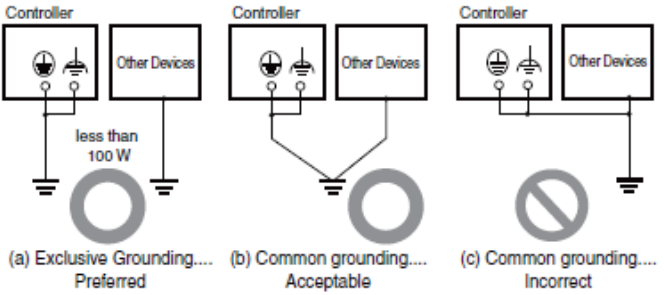
페라이트 코일을 사용하십시오. 이렇게 생겼습니다.

이것을 센서케이블 또는 네트워크 케이블 양쪽 끝에 부착하세요. 가격도 저렴하면서 노이즈 줄이는 효과는 아주 좋습니다.

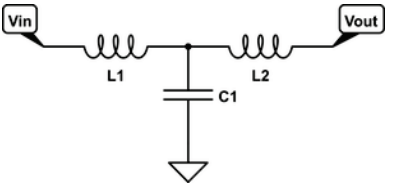


센서케이블, 네트워크 케이블의 길이를 줄이십시오. PLC 와 연결되는 I/O 선의 길이를 줄이는 것만으로도 큰 효과가 있습니다. 선이 길면 길수록 안테나 역할을 하기 때문에 노이즈를 일부러 수집하는 꼴이 됩니다.

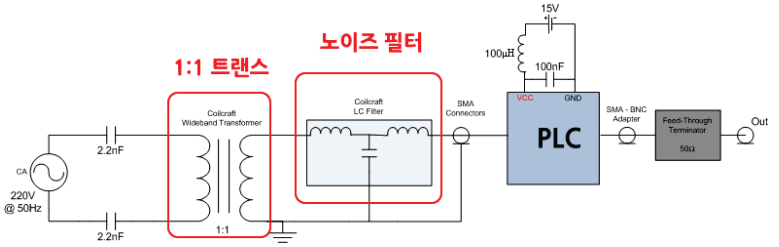
접지는 사진에서 보시는 것처럼 각각 접지하던가, 한점으로 모아서 (스타포인트라고 부른다고 합니다.) 접지하시면 됩니다. 벽에 있는 콘센트 접지선이 제대로 기능을 하고 있는지를 먼저 체크해 보셔야 합니다. 의외로 접지공사가 안되어 있는 건물이나 공장들이 많습니다



센서의 경우 센서와 컨트롤러(PLC) 사이에 L-C 필터를 사용하십시오. 필터 주파수 범위는 30dB (DC~50MHz) 규격이여야 합니다. 필터 접지는 반드시 기존 접지와 연결해야 합니다.



끝으로 전원 입력 노이즈 차단에 대한 장황한 설명이 있는데, 이 그림 한장으로 설명할 수 있을거 같습니다. 즉 입력부에 절연 트랜스를 쓰고 노이즈 필터를 달아라 이런 야그입니다.



만약 여러분이 만든 시스템이 노이즈(또는 서지)때문에 불안정하게 작동하고 있다면, 위 사항들 중 빠진게 있나 꼼꼼히 체크해 보시기 바랍니다.

접지 제대로 하는 법

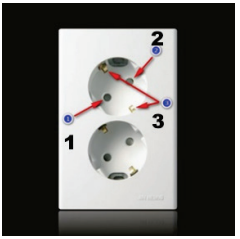
바로 앞에서 노이즈를 잡기위해서 "접지가 필수적이다" 라고 설명을 했는데요. 그럼 접지는 어떻게 해야 제대로 하는 것인지 한번 알아보겠습니다.

모든 건물이나 공장에 있는 콘센트에는 아래 사진에서 보시는 것처럼 접지하는 부분이 있습니다.

그런데 의외로 이 콘센트에 있는 접지가 제대로 접지기능을 못하는 경우도 많습니다. 오래된 건물이라던가, 건물공사시 접지를 제대로 하지 않은 경우, 인테리어 공사시 (특히 칸막이 벽에 있는 콘센트) 접지를 연결하지 않은 경우등등 많은 경우의 수가 존재합니다.

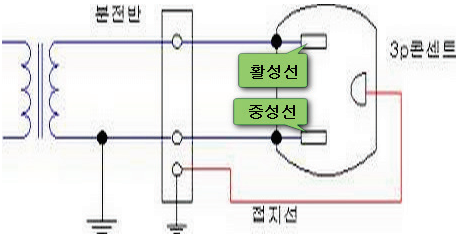
그래서 일단 콘센트의 접지가 제대로 기능을 하고 있는지 체크를 해봐야합니다. 시중에는 접지를 테스트하는 별도의 제품을 판매하고 있는데, 사용법이 매우 쉽습니다

꼽아봤을때 불이 3 개 다 들어오면 접지가 제대로 기능을 하고 있는 겁니다. 불이 2 개밖에 안들어온다면 접지가 안된 콘센트입니다. 테스터 (멀티미터)기로 확인할 수도 있습니다.



멀티미터를 AC 전압 측정에 셋팅하고, 1 번 2 번을 측정하면 220V 가 나옵니다. (반드시 장갑을 끼고 측정하세요. 잘못하면 감전됩니다. @.@)

이 상태에서 1 번과 3 번을 측정하거나, 2 번과 3 번을 측정하면 둘 중 하나에서 220V 가 표시되어야 정상입니다. 둘 중 하나는 0V 에 근접한 전압이 표시되어야 접지가 제대로 된 것입니다.

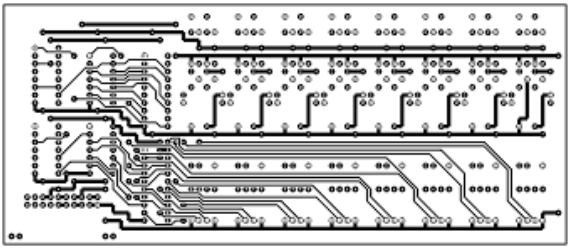


위 사진을 보면, 접지가 제대로 되었을 때 왜 그렇게 전압이 측정되는지 알 수 있습니다. 접지가 안되어 있는 경우는 둘 다 0V 로 표시되거나, 둘 다 100V 근처(중간값)로 표시됩니다. 이 경우 전기기사를 불러서 분전반을 살펴보거나, 심한 경우 건물 지하에 접지봉을 다시 손봐야 하는 경우도 생깁니다. 공사가 커질 수도 있어요. ㅠㅠ

제 4 장. 보드설계와 양산

노이즈 대책의 시작은 PCB 배선

PCB 아트웍 배선시 주의할 점을 정리해 보았습니다.



1. 배선길이는 짧을 수록 좋습니다.
배선의 길이가 길수록 L 값이 높아지고, 이로 인해 임피던스도 높아집니다. 저주파 신호는 상관없지만, 고속신호의 경우 노이즈에 취약해집니다. 그래서 배선길이는 짧을 수록 좋습니다.

2. 파워배선은 두껍게..
무턱대고 두껍게 하라는 뜻은 아닙니다. 필요한 전압과 전류를 파악하고 이를 토대로 굵기를 정해야 하는데, 이렇게 세심하게 하기는 사실상 어려우니까, 일반적으로 신호선의 3~5 배 정도로 하시면 됩니다.

만약 소비전류가 1A 가 넘어간다면, 특단의 대책이 필요합니다. 패턴도 두꺼워야하고, 그것가지고도 부족하다면, 패턴을 코팅하지않도록 해서 납땜시 납이 일부러 더 묻도록하는 배려(?)를 해주세요.

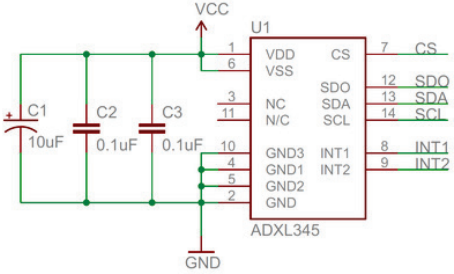
3. 루프는 형성하지 말것.
배선을 폐루프로 만들면 안테나 역할을 하기 때문에 금물입니다.

4. VIA 를 최대한 적게하자.
5. 중요한 신호선은 적당히 이격시키자.
6. 각 층은 배선방향은 격자가 되도록 하자.

노이즈 대책 필수부품, 디커플링 콘덴서

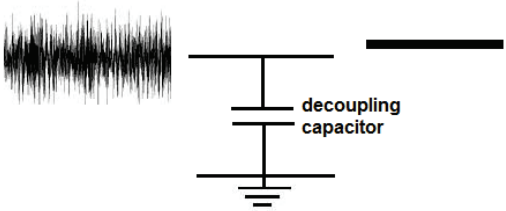
바이패스 콘덴서(혹은 디커플링 콘덴서라고도 부름)의 역할은 무엇일까요?

우리가 PCB 를 설계할때, 칩 전원단에 콘덴서 0.01uF ~ 0.1uF 정도를 배치하라고 선배들로부터 듣고 그렇게 해왔는데, 왜 이렇게 바이패스 (디커플링) 콘덴서를 배치해야만 하는걸까요?



그 이유는 노이즈 때문입니다.

칩은 전선에 신호를 보내는데, 전선에는 인덕턴스가 있기 때문에 이때 전원이 크게 저하됩니다. 결국 칩의 동작상황에 따라서 전원이 오르락 내리락을 반복해서 이것이 노이즈의 형태가 되고 맙니다.

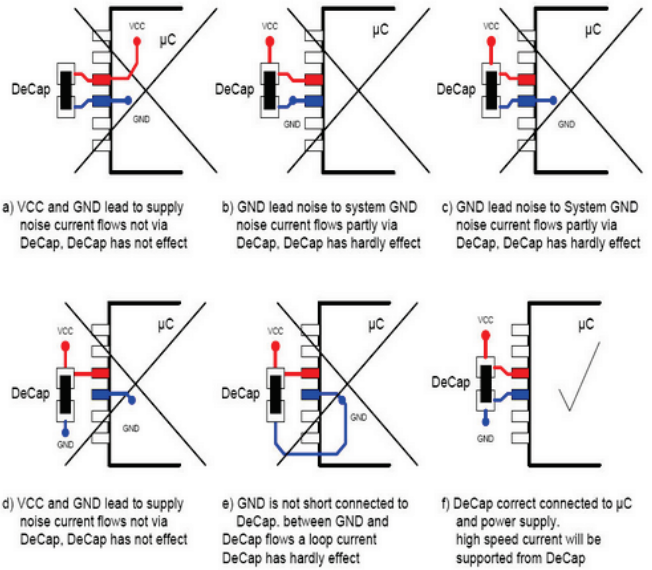


바이패스 (디커플링) 콘덴서를 달아주면, 칩이 동작할때 콘덴서에 축적되어 있던 전하를 일시적으로 칩에 공급해주는 역할을 수행하기 때문에, 결과적으로 전원이 안정되게 되는 것입니다.

이 상황을 댐에 비교해 볼 수 있는데요. 댐이 있다면, 홍수가 발생했을때, 댐에 물을 저장해둘 수 있죠. 반대로 가뭄이 발생했을 때에는 댐에 있는 물을 끌어 쓸 수 있습니다. 이처럼 바이패스 (디커플링) 콘덴서가 댐처럼 전하를 일시적으로 보관하고

있다가 필요한 상황에 즉각적으로 공급해주어서, 전원이 흔들리는 것을 막아주는 것입니다.

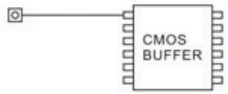
그럼 바이패스 (디커플링) 콘덴서는 어디에 다는게 최적의 위치일까요? 아래 그림 한장으로 모두 설명이 됩니다.



f) 그림처럼 칩의 전원단자와 가장 가까운 곳에 가장 짧은 경로(굵은 배선)으로 연결해야 하고, 전원이 공급되기전에 콘덴서를 거쳐서 들어가야 합니다. 그렇지 않으면 효과가 없습니다.

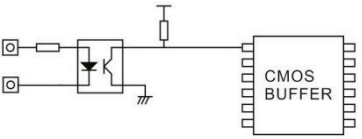
노이즈에 강한 디지털 입력설계

외부로부터 디지털 입력을 받는 회로를 한번 생각해보겠습니다.

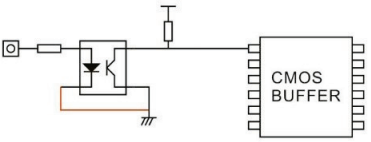


칩의 I/O 포트를 외부로 그냥 연결하는 분들도 많습니다. 대단히 잘못된 설계입니다. 노이즈는 전압은 높은 반면 전류는 미약합니다. 하지만, CMOS 칩의 기초소자인 FET 는 전압만으로 동작이 됩니다. 이렇게 입력을 받는다면 미약한 전류의 노이즈가 들어와도 진짜 신호와 구분할 수 없게 됩니다.

그래서 옵토커플러를 사용한 전원분리 (Isolation)를 많이 합니다.



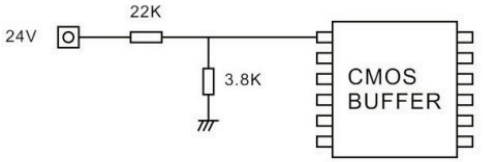
이렇게 하면, 옵토커플러에 최소 9mA 의 전류를 흘려주어야, 내부에 있는 LED 에 불이 들어와서 그 결과 CMOS 칩으로 입력 상태가 전달됩니다. 옵토커플러는 전류와 전압을 동시에 공급해야 한다는 것만으로 매우 훌륭한 노이즈 대책이 됩니다.



그럼 전원 분리를 하지 않고 옵토커플러만 사용하면 어떨까요? 옵토커플러 양단의 그라운드를 묶어버리면 됩니다. 이것도 아무것도 없는 것보다는 훨씬 좋습니다. 전류와 전압을 동시에 공급해서 LED 에 불이 들어와야, CMOS 칩으로 상태가 전달되니까요.

이렇게 되면, 비싼 옵토커플러를 구지 쓸 필요가 없습니다.

그냥 저항 2 개만으로도 전류와 전압을 동시에 공급하는 회로를 만들 수 있습니다. 단 입력전압이 5V 보다는 높아야 합니다.

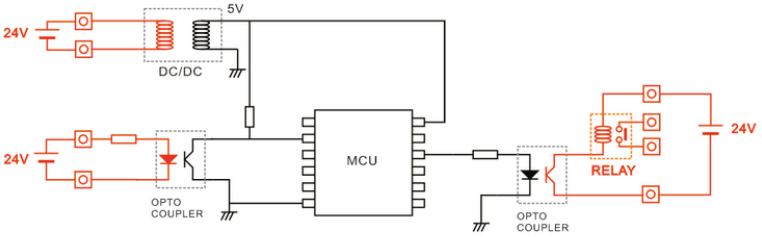


24V 입력시 22K 저항과 3.8K 저항으로 분압하여 주면, 0~5V 로 변환됩니다.

기억해두세요. 노이즈는 전류가 미약하므로, 전압과 전류를 동시에 공급해야 인식하는 입력회로를 구성한다면 노이즈를 잡을 수 있다는 것을요.

아이솔레이션이란?

전원분리를 아이솔레이션 (isolation)이라고 부릅니다. 옵토커플러만 사용하면 전원분리가 된다고 생각하시는 분들이 있습니다. 아이솔레이션은 아래 회로처럼 전원, 입력회로, 출력회로가 모두 분리가 되어서, 마치 섬처럼 5V 쪽이 완전 분리되어야 합니다.



DC/DC 컨버터는 그라운드가 분리된 정전압을 출력해주므로, 전원 분리회로에서는 필수적으로 필요합니다. 시중에는 입력측과 출력측 그라운드 분리가 안된 허접(?)한 형태의 DC/DC 컨버터도 있으므로 주의해야 합니다. 아래 사진처럼 생긴 좀 비싼(?)걸 고르면 됩니다.

여담이지만, 노이즈 대책으로 아이솔레이션이 좋은지, 아니면 아이솔레이션 없이 바리스터나 TVS 다이오드, ESD 프로텍션 버퍼칩 등의 보호소자를 사용해서 적극적으로 노이즈를 방어하는 회로가 좋은지는 전문가들 사이에서도 의견이 분분합니다.

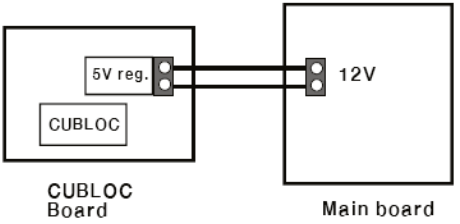
아이솔레이션은 비용이 좀 많이 드는 편이므로, 여러분이 처한 상황에 맞추어서 선택하시면 될 것 같습니다.

보드 설계시 주의할점

큐블록 또는 MCU 를 이용한 보드 설계시 다음과 같은 사항을 지켜주어야, 최소한의 노이즈 (또는 써지)로 부터 안전한 보드를 설계할 수 있습니다.

1. 큐블록 코어 (또는 MCU)가 장착된 보드에서 5V 가 생성되어야 합니다.

외부로부터 5V 를 가져오는 경우, 전원선이 안테나역할을 해서 각종 노이즈와 함께 전원이 공급되고 맙니다. 간혹 보드 2 개로 설계하시는 분들이 메인보드에서 5V 가 있다고 해서, 그걸 쭈~욱 끌고가서 MCU 까지 공급하시는 분들이 있습니다. 이건 절대 피해야 되는 상황입니다.



위 그림처럼 설계하십시오. 12V 또는 24V 를 굵은 선으로 끌어와서, MCU (또는 큐블록코어)가 있는 보드에서 5V 로 레귤레이션 하십시오. 그리고, 메인 전원공급선은 반드시 굵은 선을 사용해야 합니다. 그 선을 통해서 모든 전기가 공급되기 때문이죠. 적어도 AWG16~12 사이의 선을 사용하시기 바랍니다.

2. 메인 전원인 5V 가 케이스 바깥으로 나가도록 하지 마세요.

반대로, 보드에서 만들어진 5V 가 보드 케이스 바깥으로 나가도록 해서도 안됩니다. 이 5V 는 메인칩의 동작전원이므로 매우 소중(?)합니다. 이 전원선이 그대로 케이스 바깥으로 나간다면, 노이즈가 그 경로를 타고 들어와 메인칩의 동작에 영향을 주게 됩니다.

3. 크리스털 보다는 오실레이터를 달아주십시오.

MCU 발진소자로 크리스털을 많이 쓰시는데요. 크리스털은 수동소자이므로 서지의 공격으로 발진이 흔들릴 가능성이 있습니다. 스스로 발진하는 능동소자인 오실레이터를 쓰면 안정적인 발진파형을 MCU 쪽으로 공급할 수 있습니다. 저희 큐블록 제품군은 오실레이터를 발진소자로 사용하고 있습니다.

크리스털

오실레이터

4. 큐블록 코어 (또는 MCU)의 핀을 보드 바깥으로 그냥 노출시키지 마세요.

MCU 또는 큐블록 코어를 가지고 PCB 설계하실 때 가장 흔히 하는 실수입니다. 큐블록 코어도 결국 그 안에 MCU 가 들어있는데, 이 핀들이 보드 바깥으로 그냥 노출이 되어 있다면, 그쪽으로 노이즈나 서지가 타고 들어와 MCU 가 망가질 수 있기 때문입니다.

보호저항과 바리스터, TVS 다이오드등을 사용해서 프로텍션을 해주거나 아니면 버퍼칩을 달아주세요. 요즘 나오는 버퍼칩은 기본적으로 ESD 프로텍션 (전기충격 보호장치)이 되어 있습니다. 버퍼칩을 쓰는 것만으로도 메인 칩을 보호할 수 있습니다. 심한 경우 버퍼칩만 망가지니까, 버퍼칩만 교체하면 다시 쓸 수 있습니다.

다음은 대표적인 버퍼칩인 74HCT245 의 스펙입니다. 보시는 것처럼 ESD 프로텍션 기능이 내장되어 있습니다.

74HC245; 74HCT245

Octal bus tranceiver; 3-state

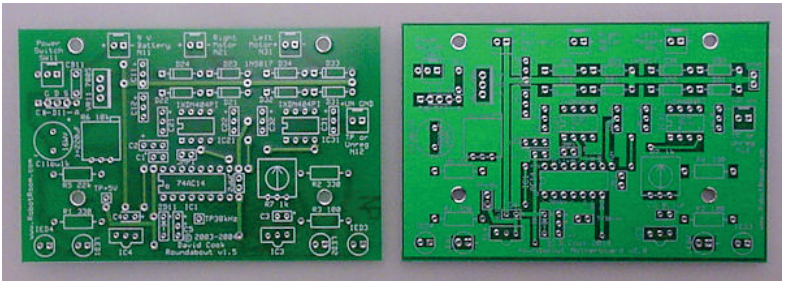
Rev. 03 — 31 January 2005 Product data sheet

2. Features

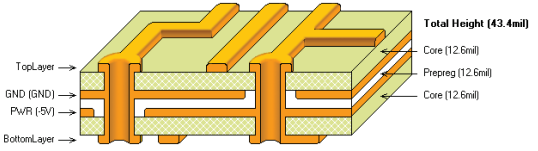
- Octal bidirectional bus interface
- Non-inverting 3-state outputs
- Multiple package options
- Complies with JEDEC standard no. 7A
- ESD protection:
  - HBM EIA/JESD22-A114-B exceeds 2000 V
  - MM EIA/JESD22-A115-A exceeds 200 V
- Specified from −40 °C to +85 °C and from −40 °C to +125 °C

5. PCB 그라운드 카파를 넓게 하거나, 아니면 4 층기판으로 하세요.

PCB 를 그릴 때, 안 쓰는 공간은 GND 카파로 해두세요. 노이즈 차단에 좋습니다.



가능하면, 2 층기판보다는 4 층기판으로 하셔서, 가운데 2 개층 중 한층을 전부 그라운드층으로 하시는게 좋습니다. PCB 내부에 그라운드 카파가 전부 깔리므로 가장 베스트 상황입니다. 비용 문제가 중요한 이슈 (예를 들면 대량 양산품)가 아니라면, 4 층기판으로 하는게 여러모로 좋습니다.

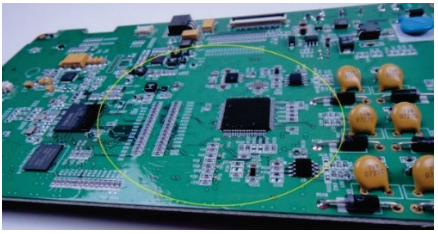


6. 패턴 간격을 벌리세요.

패턴간의 간격도 신경써야 합니다. PCB 제작공정을 보면 부식공정이 있습니다. 이게 뭐냐면 패턴을 제외한 나머지 부분을 화학물질로 녹여내는 공정입니다. 패턴사이의 간격이 너무 좁다면, 녹여내는 과정에서 미처 다 녹지못하는 문제가 발생할 수 있으므로, 불량의 원인이 됩니다. 적어도 각패턴의 두께보다는 더 벌리는 것이 좋습니다.

그 외 주의해야할 것들

기판과 습기는 상극입니다. 비닐하우스, 선박, 양어장, 바닷가, 또는 낮과 밤 온도차가 심한 지역에서 쓰이는 보드라면, 반드시 PCB 코팅제인 몰딩 에폭시를 사용하여, 습기를 차단하십시오.



보드전체를 코팅할 필요는 없습니다. 결로가 예상되는 면의 부품 중 QFP 타입처럼 핀간격이 촘촘한 곳만 코팅하면 됩니다. 투명색이라 사진에 잘 나오진 않았지만, 위 사진은 가운데 칩 주변을 코팅한 보드입니다. 너무 심하게 도포하다가 커넥터에 묻으면 접촉불량이 나오니 조심하시구요.

과열이 예상된다면, 팬을 설치하세요.

여름철 밀폐된 차안 온도는 60 도 이상까지 올라갑니다. 만약 그런 환경에서 동작되는 보드라면, 어떻게 열을 식혀줄 것인가를 고려해주어야 됩니다. 발열 부품에는 방열판을 부착하고, 통풍구를 확보해주십시오. 가끔 완전 밀폐된 케이스안에 전자기판을 넣는 분도 본적이 있습니다. 이런 시스템은 언젠가는 반드시 문제가 발생합니다.

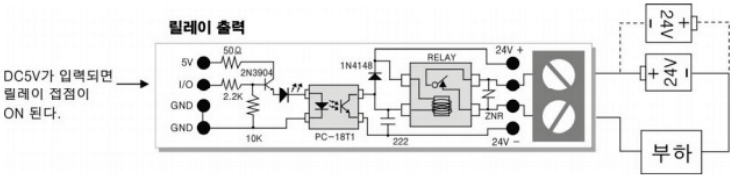
필요하다면, 팬을 설치하십시오. 팬으로 내부의 뜨거운 열기를 바깥으로 배출해주면, 한결 안정적으로 동작하게 됩니다.

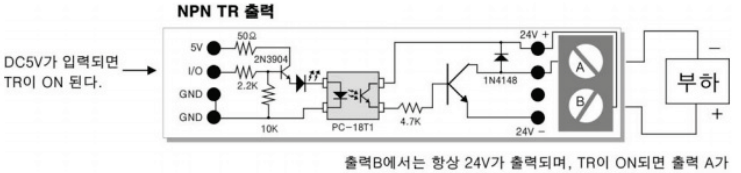
콘트롤 회로를 트랜스(또는 모터)와 가깝게 배치하지 마세요.

트랜스 (접촉기와 같이 큰 코일이 있는 부품류, 모터등 동력부)는 동작시 많은 전자파와 노이즈, 스파크등을 발생시킵니다. 이러한 노이즈원과 가까운 곳에 MCU (또는 큐블록 코어) 와 같은 콘트롤 회로를 배치한다면, 아무래도 안좋겠죠. 가급적 멀리 떨어뜨려서 배치하시고, 가능하다면 별도의 보드로 하는 것이 좋습니다.

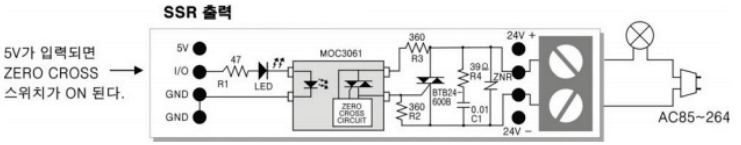
산업용 입출력 회로샘플

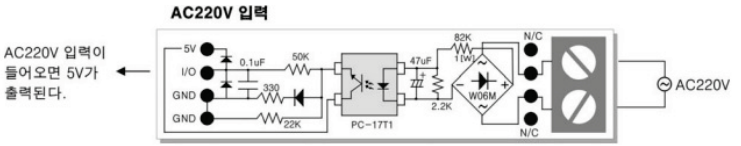
실제 산업현장에서 쓰이고 있는 입출력 회로입니다.

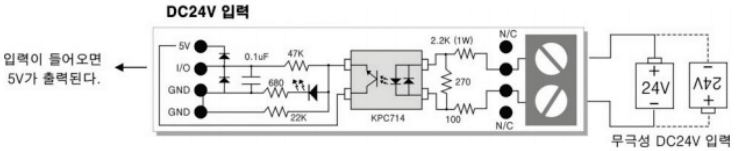




출력B에서는 항상 24V가 출력되며, TR이 ON되면 출력 A가
24V 마이너스 측과 연결된다. (부하연결시 극성 주의)







노이즈에 약한 부품은 없습니다

제가 과거 MCU 관련업계에서 일할 때, 고객한테 가장 많이 듣던 불만 중 하나가 "이 MCU 는 왜 이렇게 노이즈에 약하냐? " 였습니다.

요즘도 큐블록 코어를 가지고 보드 설계하신 분들이 "큐블록 불량"이라면서 전화주시는 경우가 많은데요. 그런 경우, 해당 큐블록을 저희가 만든 베이스보드에 꼽으면 동작이 잘 됩니다. 그럼 뭐가 문제일까요?

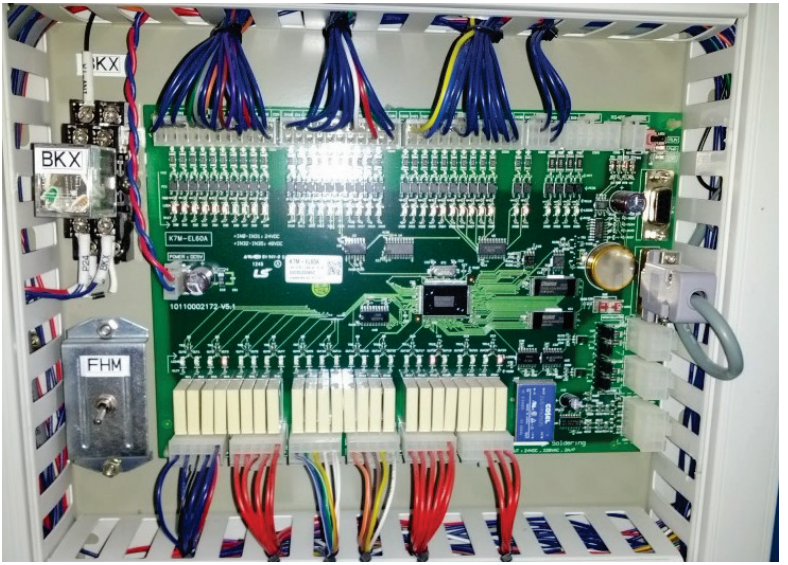
사실 MCU (또는 큐블록) 자체는 노이즈 대책을 세울 부분이 없습니다.

노이즈 대책은 PCB 와 MCU 주변소자, 그리고 전원 유입되는 부분 등의 외부적인 부분에서 하는 것입니다. **즉, 노이즈가 MCU 또는 큐블록 코어에 도달하기전에 차단해주어야 합니다.**

더 나아가, 발열대책설계, 외부 EMI 로부터 보호하기위한 금속케이스, 쉴드선 사용, 접지 등을 모두 신경 써 주어야 안정적으로 동작하는 시스템을 꾸밀 수 있습니다.

안정적으로 잘 동작하는 보드를 보면, 노이즈에 대해서 신경을 많이쓴 보드들입니다.

같은 MCU 인데, 누구는 "노이즈 문제가 없다"고 하고 누구는 "노이즈가 많이 탄다"라고 하고 있습니다. 여러분도 조금 더 검색하시고, 공부하셔서 노이즈에 강한 시스템을 꾸미는 고수가 되시길 바라겠습니다.



이 보드사진은 인터넷에서 찾은 사진인데, 노이즈에 신경 많이쓴 보드임을 한눈에 알 수 있습니다. 입력 포트는 아이솔레이션 해서, 외부 노이즈로부터 근본적으로 차단시켜 놓았습니다.

출력은 릴레이로 되어 있고, 4 개마다 커먼(공통)시켜 놓았군요. 이것도 중요합니다. 커먼이 여러개로 잡혀있으면, 그 커먼단자 하나에 많은 커런트가 흘러서, 견딜 수가 없는 상황도 생깁니다.

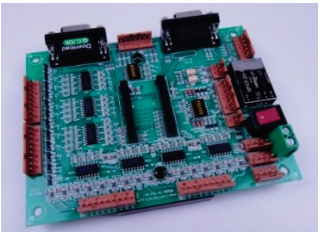
파워는 DC/DC 컨버터를 달아 놓았습니다. DC/DC 모듈은 입력측과 출력측 그라운드가 분리되어 있어서 아이솔레이션 설계시 필수입니다. 간혹 I/O 를 전부 아이솔레이션 해놓고, 전원을 일반 레귤레이터를 쓴 회로를 본적이 있습니다. 이렇게하면, 힘들여 아이솔레이션 한게 다 허사가 되고 맙니다.

보드를 크게 만들고 선도 여유롭게 배치하여, 전반적으로 잘 설계된 보드입니다. 보드내 공간이 여유가 있어야, 발열 소자들 간의 간격도 벌릴 수 있고, 그래야 공기흐름이 원활해서 냉각에도 도움을 줍니다.

큰 용량의 릴레이 하나가 별도로 달려있군요. 릴레이가 구동할 부하의 전류량을 따져보아야 합니다. 부하가 너무 커서 릴레이가 감당할 수 없을 경우에는 릴레이 융착이 발생합니다. 릴레이 접점이 녹아서 붙어버리는 현상이지요.

부하의 용량에 맞추어서 릴레이를 선택하는 것도 매우 중요합니다.

사용하지 않는 I/O 핀도 신경 써주셔야 합니다. 두가지 중 하나로 처리하세요. 입력으로 설정하고, 그라운드랑 연결하시던가, 출력으로 설정하고 LOW (0V)를 출력하시던가 하세요. 입력상태 (High 임피던스)로 핀을 Open 시켜 놓는 것은 금물입니다. 그리로 노이즈가 타고 들어가요.

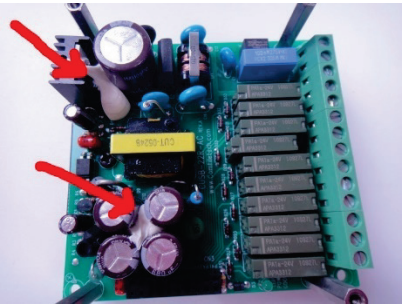


이 사진은 저희 회사 제품인 CUBASE-32M 입니다. 이 보드 역시 기본적인 부분을 지켜가면서 설계가 된 보드입니다. I/O 는 옵토커플러를 사용하여 전원 분리등을 하였고, 전원관련 패턴은 굵게 배선하였습니다.

여러분들도 PCB 를 설계하실때 이와같은 사항을 반드시 신경 써 주시기 바랍니다. 이러한 고민없이 바로 사용할 수 있는 제품이 바로 MSB 제품군입니다. 저희 회사의 오랜 노하우가 농축된 제품이므로 안심하시고 산업현장에 바로 응용 가능합니다. (CE, KCC 인증도 획득한 제품입니다.)

보드에 실리콘 쏘는 이유

시중에 유통되고 있는 보드를 보면 아래 사진처럼 실리콘이 도포되어 있는 경우를 본적이 있으실 겁니다.



이 실리콘의 용도는 무엇일까요? 크게 2 가지 목적이 있습니다.

하나는 진동 때문인데요. PCB 와 부품이 진동으로 떨어지는 것을 막기 위해서입니다. 납땜이 되어 있는 콘덴서를 손으로 잡아 땔려고 해도 안떨어지는데, 아무리 진동이 있다고 해서 저 콘덴서가 떨어질까요?

네 떨어집니다. ㅠ

하루 24 시간, 365 일 동작되는 보드가 수년간 계속 진동에 시달린다고 했을 때, 떨어집니다. 믿기 힘드시겠지만 사실입니다.

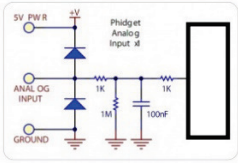
다른 하나는 이물질이 들어갈까봐 그렇습니다. 예전에 제가 전기밥솥을 만드는 회사의 MCU 펌웨어를 개발해준 적이 있습니다. 그쪽 엔지니어가 열심히 실리콘을 바르고 있길래, 구지 이렇게까지 도배를 할 필요가 있냐고 물었더니 이렇게 말하더군요.

바퀴벌레가 이 사이로 비집고 들어와서, 부품하고 보드가 타버린 채 A/S 가 들어온다구요. ㅠㅠ 고압이 발생되는 곳에는 이 물질이 접근하지 못하도록 하기위해서도 실리콘을 쏜다고 합니다.

아날로그 입력회로

큐블록 코어에는 A/D 입력 포트가 있습니다. 이 A/D 포트를 센서랑 직접 연결하시면 안됩니다. (^^;)

왜냐하면, 센서로부터 오는 노이즈나 서지 등으로 인해, 큐블록 내부에 있는 MCU 칩에 데미지가 올 수 있기 때문입니다. 최소한의 보호회로가 필요합니다.

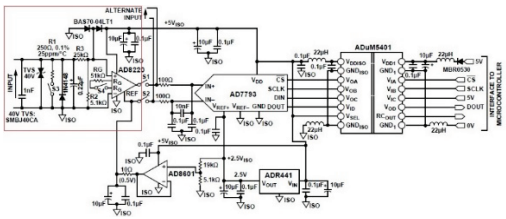


다이오드 2 개의 역할은 0~5V 를 초과하는 전압을 잘라주는 것입니다. 이러한 역할을 하는 다이오드 구성을 클램핑 다이오드라고 합니다.

MCU 는 자신의 전원전압을 초과하는 전압입력에 매우 취약합니다. 그래서 이것을 막아주는 것이죠. 그 뒤에 있는 콘덴서와 저항은 노이즈를 줄여주는 기능을 합니다.

위 회로는 센서가 0~5V 출력을 하는 경우를 가정한 것입니다. 안타깝게도 시중에 판매하고 있는 센서는 0~20mA, 0~10V, -10~+10V 등 매우 다양한 출력을 가지고 있습니다. 이러한 출력을 0~5V 로 바꾸기 위해서는 OPAMP 가 필요합니다.

게다가 보호회로와 전원 분리등을 고려하시면 회로가 한도없이 복잡해집니다. 아래 사진은 구글에서 Analog input circuit 으로 검색한 회로도입니다. 엄두가 안나는군요.



그래서 저는 다른 방법을 추천합니다. RS232 또는 RS485 로 측정결과를 출력하는 산업용 A/D 컨버터 모듈을 사용하는 것입니다.

그러면, 그 모듈안에서 골치 아픈 과정들은 모두 해결되고, 여러분은 편하게 결과만 얻을 수 있게 됩니다. 큐블록의 RS232 통신 기능을 이용해서, 데이터만 읽어오면 됩니다.

A/D 컨버터 모듈은 저희 회사의 MODPORT 라는 제품이 있습니다. A/D 이외에도 온도입력, D/A 출력도 가능합니다

아래 그림처럼 하나의 모듈로 되어 있는 제품도 있습니다.

A/D 입력이 흔들린다고 전화하시는 분이 가끔 있습니다. A/D 포트에 도달한 신호가 흔들린다면, 그 값을 그대로 읽어서 보여주기 때문에, 당연히 흔들리는 값이 표시될 것입니다. A/D 포트에 도달하기전에 노이즈를 제거해주어야 안정적인 값을 얻을 수 있습니다. 그래도 흔들린다면, 소프트웨어적으로 여러 번 읽어서 그 값을 평균내는 방법을 사용하시면 됩니다. 노이즈를 완벽히 제거할 수는 없기 때문에, 이 방법도 A/D 입력시 자주 사용하는 테크닉입니다.

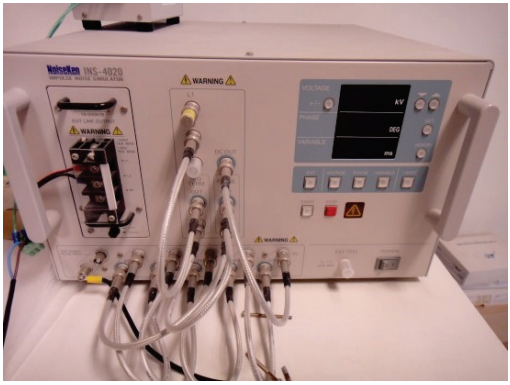
완성된 보드는 환경실험을 해봐야 합니다

여러분이 큐블록 코어 (또는 MCU)로 만드신 보드는 사무실 또는 연구소에서는 잘 돌아갈 것입니다.

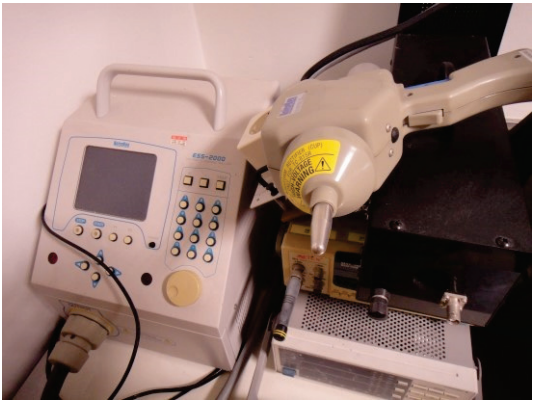
사무실은 상온/ 적절한 습도/ 무진동 상황이니까요. 하지만 산업용으로 사용되는 보드는 각종 열악한 상황에 노출됩니다. 전원선과 I/O 선에 노이즈나 서지(번개, 낙뢰), 정전기가 들어오는 경우도 있구요. 차량에 설치되는 보드라면, 하루 종일 진동에 시달릴 것입니다.

습기가 많은 환경에서 쓰이는 보드라면, 습도가 높을테구요. 결로가 맺히기도 할 것입니다. 여러분이 만든 보드에 이런 상황이 닥쳐도, 문제없이 동작할 것이라고 자신하십니까?

그래서 환경 실험을 해봐야 합니다. 저는 보드 출시전에 다음과 같은 실험을 합니다. 우선, 전원에 노이즈를 실어주는 "노이즈 발생 장치"로 노이즈 환경에서의 동작여부를 실험합니다. (수천 KV 를 노이즈를 마구마구 때리는 장비죠. ㅋ)



서지를 발생시키는 장비로, 번개, 낙뢰 충격시 동작여부를 실험합니다. 낙뢰를 맞았을때, 보드가 리셋되는 것은 상관없습니다. 멀펑션 (아무것도 안하는 돌덩이 상태)으로 빠지는 지 여부를 관찰합니다.



그 다음 고온과 저온에서 잘 동작하는 확인합니다. 열 충격 챔버를 사용해서 테스트 합니다. (고온에서 저온으로 빠르게 전환시키면서 동작여부를 테스트)

챔버에서 한가지 테스트를 더 합니다. 습도 테스트인데요. 습도를 80%이상까지 올려서 결로가 맺히는 상황에서도 동작하는 여부를 관찰합니다. 보통 48 시간 돌려보는데, 계속 쳐다보고 있을 수는 없으니까, 모니터링하는 장치를 바깥에 만들어두고, FAIL 여부를 카운트하도록 하고 있습니다.

그리고 진동테스트를 합니다. 움직일 가능성이 있는 부품류 (코일, 콘덴서등)은 실리콘으로 사전에 고정시켜놓고 합니다.

어떤 분은 발안마기로 하시는 분도 있더군요. ㅋ 저는 진동테스트기를 장만 했습니다.

이정도까지 해서 PASS 된다면, 어느정도 안심할 수 있습니다.

하지만, 지금까지 테스트한 것은 시료를 가지고 한 것이지, 양산품을 전수검사를 한게 아닙니다.

양산품 테스트는 별개 문제입니다. 저희는 제품 출하시에 전량, 챔버실 (온도 45~50 도)에 넣어서 48 시간 버닝 테스트를 한 뒤, 양품만 출하합니다.

테스트하는 동안에도 가만히 켜놓는 건 아니구요. ON/OFF 를 반복하면서 (지그를 사용해서) 정상동작 여부를 체크합니다.

보드 이상동작 사례

저는 지금까지 다양한 종류의 불량을 경험해봤는데요. 몇가지 사례를 정리해 보겠습니다.

1. PCB 자체 이상

PCB 회사마다 퀄리티의 차이가 있습니다. 여러분이 아시다시피, PCB 는 동판에 필림을 붙여서, 패턴을 제외한 나머지 부분을 녹여서 만듭니다. 그 위에 코팅을 하는 과정을 거칩니다. 이 과정에서 잘 녹지 않는 부분이 생길 수도 있고, 코팅불량이 발생할 수도 있습니다.

그래서 PCB 회사에서 자동검사 (BBT)를 합니다. 대충 이렇게 생긴 기계로 하는겁니다.

여러분은 PCB 를 주문하실때, 반드시 BBT 검사를 의뢰하십시오. 간혹 물량이 많아서 바쁘다는 이유로 목시검사만 해서 납품하는 경우가 있습니다. 목시검사는 눈으로 한번 스~윽 봤다는 뜻입니다. (이게 뭡니까? ㅠㅠ, 목시검사 운운하면서 이상 없을 거라고 하는 업체가 있다면 믿고 거르시면 됩니다.)

BBT 기계는 실제 CAD 데이터와 PCB 의 홀을 비교하면서, 일치하는지 여부를 검사합니다. 단선/합선 등을 정확히 찾아냅니다.

2. 납땜 이상

테스트용 보드는 직접 납땜하시겠지만, 양산은 보통 SMT 로 하시거나, 전문 임가공업체에 맡기실 겁니다. 세상 모든 병원의 의사가 명의가 아니 듯, SMT를 다루는 회사도 천차만별입니다. 작업자의 숙련정도가 품질에 영향을 주게 됩니다.

임가공 회사도 마찬가지입니다. 디핑 작업자의 숙련도, 세척공정, 드라이 공정등의 전문성여부가 제품의 품질차이를 만듭니다.

이 모든 것을 관리 감독하는 것은 여러분의 몫입니다. 이러한 전문회사에서 하는 경우에도 냉땜, 미삽입, 오삽입이 발생하는 것을 저는 자주 보아왔습니다. 매의 눈으로 감시하고 자주 방문해서 진행상황을 체크하는 등의 정성(?)을 쏟아주어야 원하는 품질의 제품을 받으실 수 있습니다. 불량발생시 뒷 감당은 여러분이 해야되니까요. 좋은 품질의 제품이 그냥 얻어지는게 아닙니다.

3. 플럭스 세척

플럭스는 보드에 납이 잘 묻으라고 바르는 화학약품입니다. 간혹가다 플럭스가 남아있는 채로 제품을 출하하는 경우도 봅니다. ㅠㅠ 이건 절대 피해야되는 상황입니다.

플럭스는 반드시 제거되어야 합니다. 부식 등 PCB 에 나쁜 영향을 주고, 주변 다른 물질과 화학반응을 일으켜 결국 전기회로에도 뭔가 영향을 주게 됩니다. (심지어 전기를 통하게 만드는 경우도 생깁니다.)

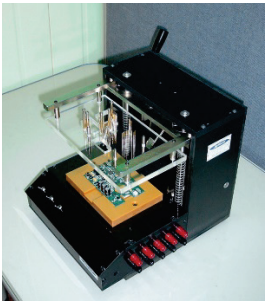
납땜이 끝났으니 원래의 목적을 다 한겁니다. 반드시 세척한후 사용해야 됩니다. 바쁘다는 이유로, 귀찮다는 이유로 SKIP 해서는 안됩니다.

시중에는 무세척 플럭스도 있지만, 저는 무세척 플럭스는 사용안하고 있습니다. 그냥 플럭스를 쓰고, 세척하면 되는거지, 비싼 무세척 플럭스를 쓰고 세척안한 끈적끈적한 상태로 놔둔다는게 영 찜찜하더군요.

지그를 사용하여 테스트하세요

보드 양산을 처음 시도하시는 분들은 보드 양산이 끝나면, 간단한 테스트후 바로 출하하는 경향이 있습니다. 생산된 보드는 각종 변수(?)에 의해서 불량이 섞여 있을 가능성이 높기 때문에 테스트를 꼼꼼히 할 필요가 있습니다.

그러려면 보드상의 여러 개의 특정포인트를 연결해야 하는데, 수많은 측정 포인트를 사람이 일일이 연결해서 테스트하기에는 무리입니다. 수량이 더 늘어날수록 더욱 힘든 일이 됩니다. 그래서 지그라는 것을 사용합니다.



지그를 만들어서 물리면, 측정포인트에 핀이 내려옵니다. 이 핀을 별도의 테스트 기판과 연결하여 동작이상 여부를 체크합니다.

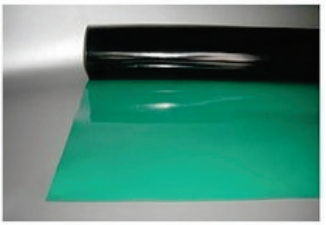
가장 기본적인 기능부터, 필수적인 기능까지 모두 테스트하여 이상 없는 보드만 출하하는 노력이 필요합니다.

지그 하드웨어는 별도 업체에 의뢰하시는 편이 빠릅니다. 직접 만들기에는 좀 무리가 있습니다. (직접 만드신다구요? 그렇다면 당신은 금손 ^^;;)

지그 하드웨어가 있다고 해서, 바로 테스트가 가능한 것은 아닙니다. 테스트 보드와 테스트용 프로그램을 만들어야 합니다. 보드 양산의 길이 멀고 험하군요 ㅠ

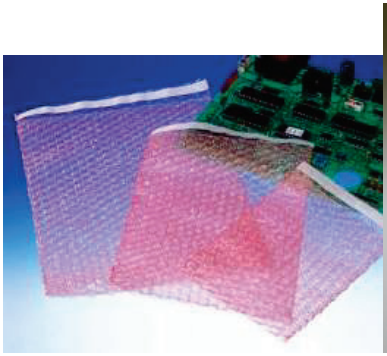
정전기 주의보

PCB 또는 보드를 다루는 작업자들이 흔히 간과하기 쉬운 부분이 정전기에 대한 것입니다. 기본적으로 작업대 위에는 정전기 매트를 깔아두세요.



정전기를 방지하기 위해서 정전기 방지장갑을 끼고, 고무재질로 된 신발 (슬리퍼)를 신고 있어야합니다. 의자도 정전기가 통과할 수 없는 프라스틱 구조물로 된 의자를 써야합니다. 작업자가 공중에 붕 떠있다면, 정전기는 발생하지 않습니다. 그럴 수 없으니 이런 방법을 쓰는 겁니다.

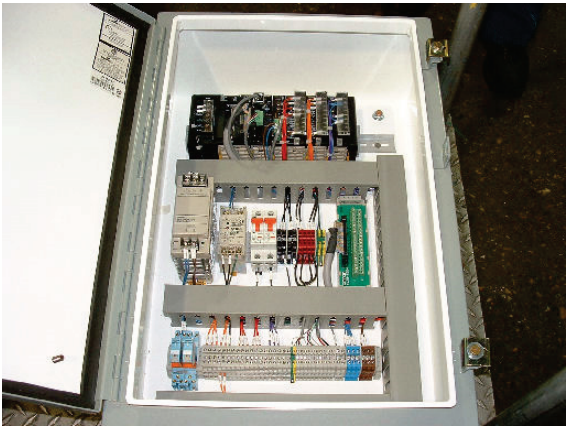
테스트가 끝난 보드는 정전기 방지 비닐팩에 포장해야 합니다.



이런 종류의 비닐팩을 Anti-Static 봉투라고 합니다. 일반 봉투보다는 조금 비쌉니다. 어렵게 생산해서, 테스트까지 마친 보드를 포장지에 넣다가, 정전기로 파손시키고 싶지 않으시겠죠?

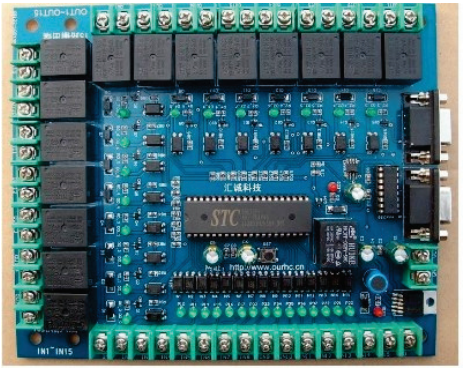
PLC 를 원보드로 만들때 주의사항

PLC 로 제품을 만들어서 양산하시는 분들이 겪는 일들을 적어보겠습니다.



PLC 로 만든 시스템이 하나 둘씩 팔리기 시작하면서, 고민에 빠지게 됩니다.

PLC 는 비싸기 때문에, 제조원가에서 차지하는 비중이 큰편입니다. 그래서 PLC 를 하나의 PCB 로 보드화하면 원가가 절감될 것이라는 단순한 생각으로 외주개발을 맡겨서 원보드로 만듭니다.



발주를 받은 외주업체에서는 MCU 를 써서 기존 PLC 의 동작을 그대로 구현합니다. 잘 돌아가는 것 같아서 바로 교체하여 현장에 투입했지만, 자꾸 불량이 나오고, 동작이 예전같지 않다고 합니다.

왜 그럴까요? 대략 3 가지의 원인이 있습니다.

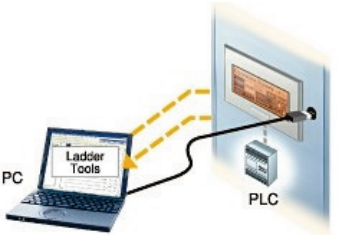
1. MCU 는 C 언어나 어셈블리어로 개발을 합니다. 레더로직이 아닙니다. 여기에서 첫번째 문제가 발생합니다.

레더로직은 스캔반복실행 구조입니다. 1 행부터 끝행까지 수마이크로초 ~ 수밀리초 간격으로 계속 스캔하면서 실행합니다. 중간에 노이즈가 들어와서 출력이 틀어진다고 해도, 다음 스캔에서 이를 바로잡아줍니다.

일반적으로 C 언어로 프로그램을 짜는 분들은 이렇게 코딩하지 않습니다. 한번 I/O 를 건드리면, 그대로 있겠지 하고 다시는 쳐다보지 않는 식으로 프로그램을 합니다. 그래서 노이즈에 의해서 I/O 상태가 변하면, 그 상태가 그대로 유지됩니다.

2. PLC 는 노이즈, 서지대책이 훌륭한 완제품입니다. 안전규격/EMI 규격도 다 PASS 한 제품입니다. 반면 여러분이 만든 원보드 PCB 는 인증도 없고 노이즈/서지대책도 완전하지 못한 경우가 많습니다. 원가 절감에 신경 쓰다 보니, 기본적인 스파크 /

서지보호 회로조차 넣지 않은 보드가 대부분 이였습니다. 이러니 필드에서 문제가 생길 수밖에 없죠.



3. 외주업체는 개발해주고 떠나버립니다. 추후 사소한 수정사항이 생기면 곧바로 적용시키기 어렵습니다. 하지만 PLC 는 수정 및 보완이 비교적 쉬운 편입니다. 필드에 노트북 하나만 들고 나가서 수정해주는 것도 가능합니다. 원보드 PCB 는 이것이 어렵습니다.

그래서 결국 다시 PLC 로 돌아오는 경우를 많이 보아 왔습니다. PLC 를 보드화 하는 것은 결코 쉬운 작업이 아닙니다.

큐블록을 사용해서 보드화 한다면, 적어도 1 번 (레더로직)과 3 번 (필드수정)문제는 해결됩니다.

2 번 (PCB 설계 과 노이즈 대책)는 여러분의 몫입니다. 보드설계 경험이 없다면 전문가의 도움을 받아서 하는게 좋습니다.

개발과 상용화는 하늘과 땅 차이(퍼온글)

개발과 개발한 제품을 상용화 했던 경험이 있던 사람이라면 당연히 알법한 내용이다. 제품을 만들기 위해 90%까지 완료했다면 상당히 잘 개발 했다고 할 수 있지만, 이것을 시장에 내논다면 100% claim 이 걸려 앞으로 벌고 뒤로 까먹는 황당한 경우가 생길 수 있다. 경우에 따라서는 시장 선점을 위해 출혈을 감소하더라도 이러는 경우는 있다.

나머지 10%는 상용화를 위해 반드시 필요한 부분인데 이 부분이 생각보다 까다롭고 오래 걸리는 부분이다. 개발은 1 개월에 끝냈는데 10%를 마무리 짓기 위해 1 년을 소요할 수 도 있거나 아예 상용화가 불가 할 수도 있다. 그만큼 아무나 사용할 수 있는 완성도가 높은 제품을 만든다는건 정말 어려운 일일것이다.

그런데, 이를 매우 쉽게 보는 사람들이 간혹 보인다. 양산이나 상용화에 대한 개념이 없는건지 이유는 잘 모르겠지만, 이렇게 얘기하는 사람은 그 사람의 경력을 떠나 아마추어로 밖에 보이지 않는다.

예를 들어 그들 표현에 의하면 개나 소나 개발할 수 있는 그런 제품도 상용화 및 양산 하려면 많은 사람들이 매달려 개발의 완성도를 높여야 될 뿐만 아니라 개발이외의 작업에 상당한 시간과 노력을 들여야 한다. 즉 개발만 한다고 될일은 아니고, 상용화는 개발과 달리 또다른 차원인 것이다.

큰 기업에서 양산을 많이 겪어 보지 않고 단기 프로젝트성으로 몇개만 만들고 마는 제품만 많이 만들어본 사람들은 그 프로젝트의 수와 상관없이 상용화에 대한 실무는 사실상 매우 부족한 편인 듯 하다.

그로 인해 난 뭐든 개발할 수 있다란 자신감과, 이미 상용화된 제품을 아무것도 아닌것인냥 취급하며 "내가 하면 이거 훨씬 더 싸고 좋게 짧은 시간에 만든다" 라는 호언 장담을 늘어 놓기도 한다. 아마도 만들 수는 있을지 모르지만 상용제품 수준은 아닐 것이다.

말을 좀 정리하자면, 소매시장에 제품으로 나왔다는건 그 제품의 기술적 수준을 떠나 엄청난 노력과 시간이 들어갔다는 사실을 알아야 할 것이고, 동병상련인 개발자나 엔지니어 출신들은 특히 같은 입장이므로 서로를 무시하는 그런 일은 하지 말아야 할 것이다.

[출처] 개발과 상용화는 하늘과 땅차이|**작성자** 캐럴리

제품선택 가이드

대량으로 양산하는 제품이라면 MCU 로 하는 것이 맞습니다. 마이크로칩, ATMEL, STmicro 등이 국내에서 가장 많이 사용하는 MCU 메이커 입니다. 요즘에는 ARM CORTEX 코어로 된 32 비트 MCU 를 많이 쓰는 추세입니다. (MCU 를 사용하려면 C 언어 프로그래밍 기술과 PCB 제작 기술이 있어야 합니다.)

준 양산품이라면 큐블록이 적합합니다. 큐블록 코어는 PCB 에 꼽을 수 있는 반도체형이기 때문에, 여러분이 직접 PCB 를 만들어서 그 곳에 큐블록 코어를 꼽아서 생산이 가능합니다.

PCB 개발, 생산경험이 있으신 분이라면, MCU 보다는 간편하게 개발할 수 있는 큐블록으로 선택하십시오. 시간이 절약됩니다. 아시겠지만 시간절약은 곧 비용절감을 의미합니다.

PCB 를 제작할 능력이 없는 분이라면, 두가지 선택이 있습니다. 우선 큐블록 MSB 시리즈입니다. 이미 제작된 모듈형태의 CUBLOC 이므로 곧바로 현장에서 설치 사용할 수 있습니다.

그리고 모아콘이 있습니다. 모듈 여러개를 조합하여 하나의 시스템을 완성하는 모듈조합형입니다.

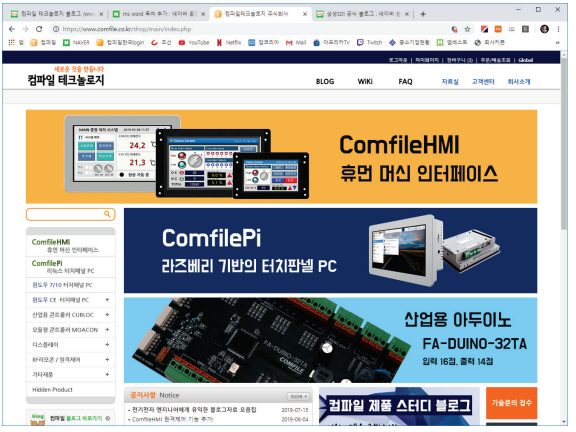
PC 베이스 급의 시스템을 계획하고 계신다면, 저희 회사 CUWIN/컴파일파이 또는 CUPC 시리즈를 선택하시면 됩니다.

만약 C#언어로 프로그램이 가능하시다면 CUWIN 또는 컴파일파이로도 충분합니다. CUWIN 은 윈도우 CE 환경으로 되어 있어 비주얼 스튜디오에서 C#으로 작성한 코드를 다운로드해서 실행하는 것이 가능합니다. 컴파일파이는 리눅스이지만 C#으로 작성한 코드가 돌아갑니다.

CUWIN 과 컴파일파이는 CUPC 보다는 저렴하고, 바이러스 걱정이 없고, OS 를 추가로 구매할 필요가 없습니다.

만약 PC 에서 작성한 소프트웨어를 그대로 실행시키려고 한다면, 어쩔 수 없이 윈도우 7, 10 베이스의 CUPC 시리즈를 선택해야 합니다. 다양한 화면크기의 PC 급 터치패널 일체형 제품이 준비되어 있습니다.

제품 구매는 컴파일 사이트에서 .... (www.comfile.co.kr)



제 5 장. 컴파일 파이

요즘 저희 회사에서 가장 핫한 제품인 컴파일 파이에 대해서 알아보겠습니다.

아직도 윈도우CE제품을 쓰신다구요?

여러분도 잘 아시다시피 윈도우 CE 는 15 년 전 OS 입니다. 닷넷언어 (C#, VB.NET)를 쓰기위해서 과거에는 윈도우 7,10 또는 윈도우 CE 말고는 다른 선택지가 없었습니다.

하지만 지금은 리눅스에서도 닷넷언어를 쓸 수 있게 되었죠. 심지어 리눅스에서는 데스크탑에서 지원하는 프레임웍을 지원합니다. (반면 윈도우 CE 는 마이너 버전인 컴팩트 프레임웍을 사용해야 합니다.)

가격도 리눅스 기반의 디바이스가 월등히 저렴합니다. 그런데 H/W 스팩은 더 좋죠.

이제 리눅스+닷넷은 새로운 흐름입니다.

저희 회사의 컴파일 파이와 AdvancedHMI 를 쓰시면, 최신 트랜드 3 가지를 모두 경험하실 수 있습니다.

1. 리눅스 기반의 터치패널 PC (라즈베리파이, 라즈비안 채용)
2. 닷넷 언어 지원 (Visual Studio 2017)
3. AdvancedHMI 지원 (드라이버와 콘트롤을 갖춘 기반 소프트웨어)

아직도 비싼 H/W 와 저성능 퍼포먼스인 윈도우 CE 디바이스에서, Visual Studio 2008 로 컴팩트 프레임웍을 사용하고 계신다면, 이제 새로운 시대의 흐름에 몸을 맡겨보시는게 어떨까요?

컴파일 파이의 장점

요즘들어 컴파일파이(ComfilePi)에 대한 문의도 부쩍늘고, 고객의 관심도 많아진 것 같습니다.

컴파일 파이는 윈도우 CE 기반의 터치패널 PC 제품을 대체할 수 있는 제품입니다. 윈도우 CE 기반의 터치패널 PC 는 15 년전부터 있던 제품이다 보니 아무래도 여러가지 면에서 최근 출시된 ComfilePi 에 뒤쳐질 수 밖에 없습니다.

현재 저희 회사는 물론 타사에서도 윈도우 CE 기반의 제품이 많이 나오고 있는데, 이 제품들과 컴파일 파이와 차이점을 한번 짚어보겠습니다.

윈도우 CE 기반의 제품은 일단 기본 CPU 가 32 비트 싱글코어 ARM 칩으로 되어 있습니다. 컴파일 파이는 64 비트 쿼드코어로 되어 있지요. 클럭주파수(스피드)도 컴파일 파이가 높습니다.

	컴파일파이	타사의 윈도우 CE 제품군 또는 CUWIN 제품군
CPU	64 비트 쿼드코어	32 비트 싱글코어
스피드	1.2GHz	500MHz ~ 1GHz

다음은 메모리를 보겠습니다. 컴파일 파이는 SD 카드를 메인 메모리로 사용합니다. 반면 윈도우 CE 제품군은 낸드플레쉬를 메인메모리로 씁니다. 낸드 플레쉬는 여러분도 아시는 것처럼 배드블럭 발생시 치명적 오류가 발생하는 단점이 있습니다. 이것을 극복하기 위해서 내부적으로 배드블럭을 피해가는 메커니즘을 갖춘 SD 카드 (또는 eMMC)를 다는 추세입니다.

다행히 컴파일파이에서는 낸드 플레쉬를 채택하지 않고 있습니다. 램 용량도 2 배이상 차이가 납니다.

	컴파일파이	윈도우 CE 제품군
메인메모리	SD 카드(최대 128G 까지)	256~512MB 낸드 플레쉬
램	1 기가	256~512MB

그 다음은 개발환경을 보겠습니다. (컴파일파이는 리눅스 기반이긴 하지만 윈도우에서 사용하는 닷넷언어도 지원합니다.)

윈도우 CE 제품군은 VISUAL STUDIO 2008 까지만 사용가능 합니다. 그 이후버전은 아예 사용할 수가 없습니다. 더 슬픈 소식은 VISUAL STUDIO 2008 은 유료이며, 상당히 고가라는 점입니다.

반면 컴파일 파이는 최근 출시된 VISUAL STUDIO 2017 을 지원하며, 무료로 다운로드 받으실 수 있습니다. (단, 마이크로 소프트 계정 가입을 해야합니다.)

그리고, 윈도우 CE 에서 사용하는 닷넷은 사실 반쪽짜리입니다. 컴팩트 프레임웍이라는 이름으로 그럴싸하게 포장했지만, 데스크탑 프레임웍을 전부 구현할 수 없어서 일부 기능을 제한시킨 마이너 버전입니다.

컴파일 파이에서는 데스크탑 프레임웍을 지원합니다. PC 에서 동작하는 닷넷프로그램이 그대로 컴파일 파이에서 돌아갑니다. 더이상 컴팩트 프레임웍에 맞추기 위해서 코드를 손볼 필요가 없어 졌습니다.

	컴파일 파이	윈도우 CE 제품군
개발환경	최근 VISUAL STUDIO 를 무료로 사용가능	VISUAL STUDIO 2008 유료
프레임웍	데스크탑 프레임웍	일부 기능이 제한된 컴팩트 프레임웍만 사용가능

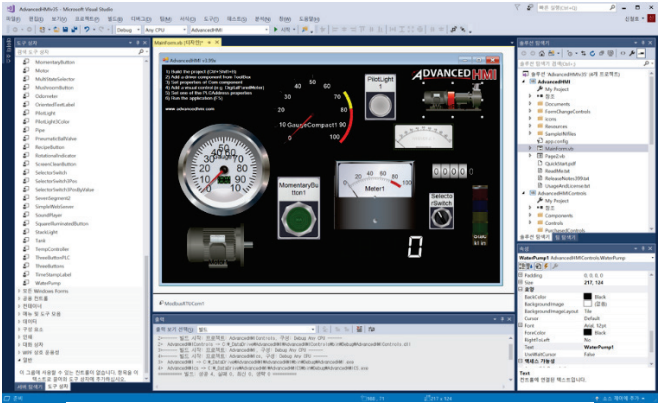
새로 프로젝트를 시작하신다면 컴파일 파이가 여러모로 유리할 것 같습니다.

AdvancedHMI와 컴파일파이는 찰떡궁합

ComfilePi 는 리눅스 운영체제의 터치 패널 PC 입니다. 여기에 비주얼 스튜디오 환경에서 C# 으로 짠 프로그램을 실행시킬 수 있습니다.

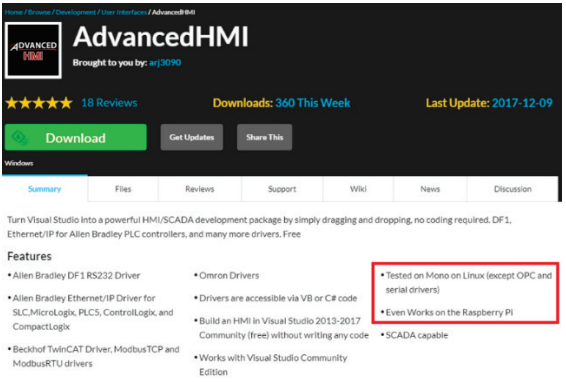
말은 쉽지만, C# 으로 동작가능한 프로그램 하나를 짜는게 그리 간단치 않습니다. 비주얼 스튜디오에서 기본적으로 제공하는 콘트롤들도 그닥 이쁘지도 않고요. 외부장치와 통신 프로토콜도 일일이 신경 써주어야 하구요.

각설하고 바로 AdvancedHMI 를 소개해 드리겠습니다. 미국 advancedhmi.com 에서 제공하는 무료 HMI 솔루션입니다.



1. 일단 무료입니다.
2. 비주얼 스튜디오 환경에서 돌아갑니다. (비주얼 스튜디오 2017 도 무료입니다.)
3. 생성된 실행파일은 원래 PC 용 실행파일인데, 이 파일이 ComfilePi 에서도 실행됩니다.

아래 다운로드 링크에 있는 설명을 보면, 라즈베리파이에서도 실행된다고 되어 있습니다.



간단한 화면구성이라면 C# 코드를 짤 필요도 없습니다. 만약 좀더 복잡한 기능을 원한다면, C# 이나 Visual BASIC 으로 코드를 추가해서 여러분이 원하는 기능을 구현할 수 있습니다.

C# 을 쓸 수 있는 분이라면, AdvancedHMI 로 기본 틀을 만들어 놓고, 커스터 마이징 해가는 방향으로 개발을 하시면 될 것 같습니다. AdvancedHMI 가 가지고 있는 기본적인 장점들이 꽤 많습니다.

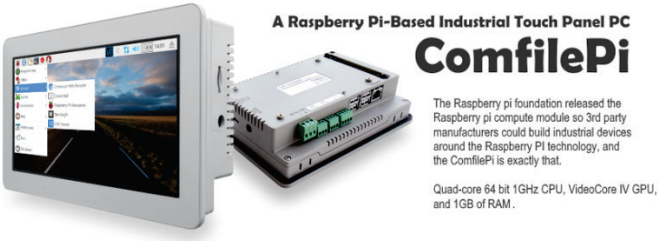
1. Allen Brandley 계열 PLC 이더넷 드라이버 지원
2. Beckhoff Twincat 이더넷 드라이버 지원
3. MODBUS RTU, TCP 드라이버 지원
4. Omron 드라이버 지원
5. 다양하고 모양도 이쁜 콘트롤들 다수(버튼, 램프, 인디게이터, 게이지등등) 지원.

산업용 라즈베리파이, 컴파일 파이 내부

라즈베리파이는 노이즈, ESD(전기적충격 즉 서지, 스파크, 정전기등)에 매우 취약한 구조로 되어 있습니다.

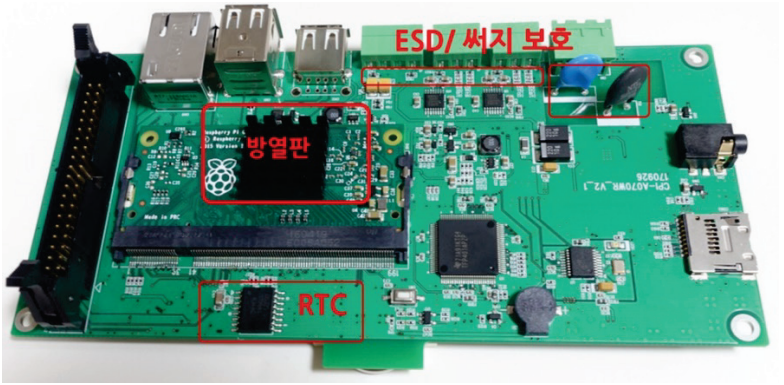
시중에서 일반적으로 판매하고 있는 터치+LCD+라즈베리파이 조합은 라즈베리파이 원래 보드를 그대로 쓰기 때문에, 노이즈및 ESD 에 취약한 단점을 보완할 수 없는 구조입니다.

컴파일파이 제품 외형입니다.

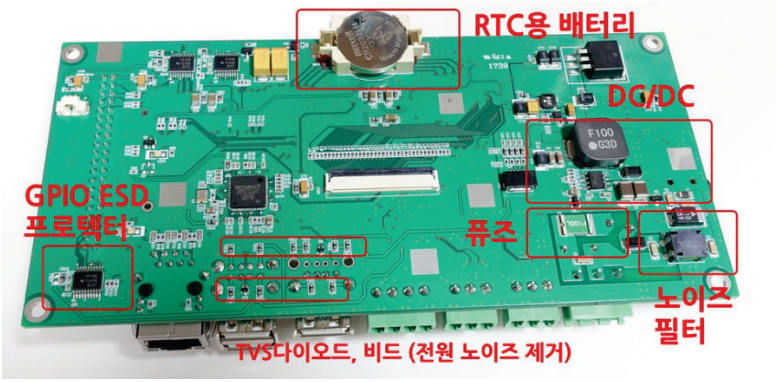


저희 컴파일 파이(ComfilePi)은 이런 류의 제품들과는 근본적으로 다른 구조를 채택하고 있습니다.

다음은 컴파일 파이의 내부 사진입니다.



방열판을 붙여서 방열대책을 세워놨습니다. RS232 등 모든 입출력포트에는 ESD 및 서지 보호 회로가 추가되어 있습니다.



40 핀 GPIO 단자에도 ESD 프로텍터를 부착해 놓았습니다. 전원 입력쪽에는 자동 복원 퓨즈와 노이즈 필터회로를 적용했습니다. 오리지널 라즈베리파이에는 없는 RTC 와 배터리도 추가했습니다. 전자파 인증(CE, FCC)도 득한 제품입니다.

산업용으로 쓰기에 손색없는 제품입니다.

제 6 장. 컴파일 HMI

저희 회사의 "휴먼 머신 인터페이스" 제품인 ComfileHMI 를 소개합니다.

그래픽 터치 개발을 앞둔 당신의 선택은?

최근 들어 그래픽 터치 인터페이스를 갖춘 기기 개발의뢰가 증가하고 있습니다.

이러한 제품을 개발하는 방법은 크게 4 가지로 생각해볼 수 있습니다.

1. Touch Panel PC 와 별도의 콘트롤러 보드를 이용하는 방법

윈도우 7/10 을 지원하는 터치패널 PC 에 InTouch, 랩뷰 와 같은 소프트웨어를 쓰고, 별도의 콘트롤러 보드와 통신하는 방식으로 시스템을 구성하는 것입니다.

장점은 비교적 빠른 개발이 가능하다는 점이고, 단점은 원가가 높다는 것입니다.

2. Window CE, 리눅스 터치패널 PC 와 별도의 콘트롤러 보드를 이용하는 방법이 있습니다.

우리 회사에서도 윈도우 CE 기반의 CUWIN 제품과 리눅스 기반의 컴파일파이 제품이 있습니다만, C#을 쓸 수 있는 엔지니어가 필요합니다. 그리고 별도의 콘트롤러는 MCU 등을 써서 만들기 때문에, 펌웨어 엔지니어가 또 필요합니다.

3. HMI + PLC 를 이용하는 방법

지금까지 많은 분들께서 흔히 이용해오던 방법입니다. HMI 는 휴먼 머신 인터페이스를 뜻하는데, 보통 터치스크린이라고 부릅니다.

HMI 는 작화툴을 이용하기 때문에, 별도의 C# 엔지니어가 필요없이 누구나 쉽게 작화할 수 있다는 장점이 있습니다. PLC 역시도 레더로직을 쓰기 때문에, 비교적 쉽게 프로그래밍 할 수 있습니다.

이 방법이 가장 좋은 솔루션처럼 보입니다만, 자동화 분야에 특화되어 있어서, 앞의 사진과 같은 전용기기나 상업기기에는 적합하지 않습니다.

4. HMI + MCU (또는 큐블록) 솔루션

현재 저희 회사에서 밀고(?)있는 솔루션입니다. HMI 로 편하게 작화를 하고, MCU(또는 큐블록)를 이용한 보드로 콘트롤을 담당하는 것입니다.

아시는 것처럼 MCU (또는 큐블록 코어)를 이용한 제어보드는 원가가 매우 저렴합니다. 그리고 양산에도 적합합니다.

HMI 는 작화가 쉽기 때문에, 별도의 엔지니어가 필요없이, MCU (큐블록) 엔지니어가 충분히 담당할 수 있습니다. 저희 회사에서 나온 ComfileHMI 는 전용기기, 양산기기에 적합한 형태 (오픈 프레임 타입도 있음.) 로 되어 있습니다.

MCU와 HMI의 만남

요즘은 조그마한 기기를 만들 때에도 LCD 디스플레이가 있고 터치가 되는 제품을 만들고 싶어 하시는 분이 많습니다.

완성된 제품이 뭔가 그럴싸해 보이는 효과까지도 가지고 있으니까요. 업체 입장에서도 경쟁에 유리할 뿐더러 제품가격도 더 받을 수 있어서 부가가치가 높기 때문입니다.

양산을 목적으로 하는 제품은 어쩔 수 없이 MCU 를 써야 하지요. 보통은 32 비트 MCU 를 쓰고 LCD 와 터치패널을 직접 연결하는 구성을 생각하게 되는데요. 어찌어찌 회로를 구성했다고 해도, 이제부터 골치 아픈일의 연속입니다. LCD 화면에 글자 하나라도 나오게 하려면, 공부해야 할 것이 산더미입니다. 폰트는 어떻게 할까요? 트루타입 폰트는 언감생심 꿈도 못 꾸고, 울퉁불퉁한 비트맵폰트라도 구해서 표시해야 합니다.

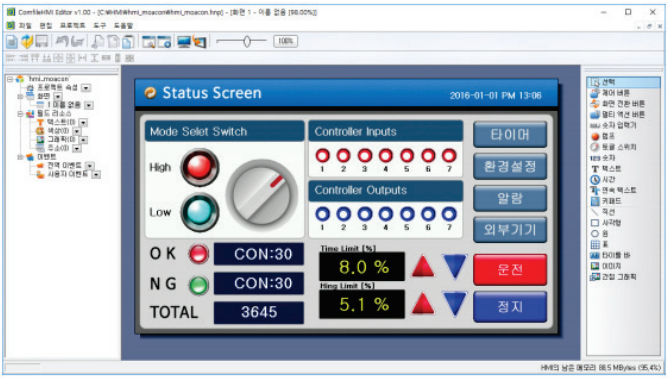
하지만, 고객은 환타스틱한 그래픽 화면을 원하고 있죠. 뭔가 부족하기 때문에 어쩔 수 없이 jpg 나 png 그래픽 파일 포맷을 공부해서, LCD 화면상에 뿌리는 방법밖에는 없어 보입니다. 이 역시도 만만한 일은 아니죠.

jpg 나 png 그래픽 파일은 용량이 큰 경우가 많기 때문에 MCU 의 플레쉬 메모리에 저장하기는 곤란합니다. SD 카드에 저장해서 불러오는 수 밖에 없습니다. SD 카드에는 FAT 이라는 파일 포맷으로 저장해야 됩니다. 그래야 PC 에서 만든 그래픽 파일을 SD카드로 옮길 수가 있습니다. MCU엔지니어는 SD카드의 FAT포맷을 해석해서 그래픽 파일을 LCD 화면에 뿌려주는 작업을 해야 하는 것이지요.

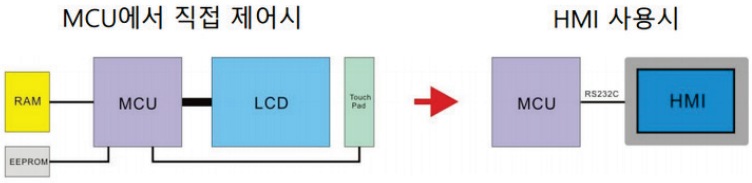
폰트작업, 그래픽 포맷분석, FAT 파일 포맷 이해 등이 우선 해결해야할 과제입니다. 생각만해도 골치가 아프시죠? 이런 고민들을 한방에 해결할 방법이 있습니다. 바로 HMI 와 MCU 를 연결하는 것인데요. 저희 회사에서 출시된 ComfileHMI 에서는 MCU 에서 쓰기 편한 통신 프로토콜인 Simple Modbus 를 지원하기 때문에 MCU 와 쉽게 연결할 수 있습니다. 심지어 소스도 드려요.

(여기서 잠깐! SimpleModbus 란? 기존 MODBUS 프로토콜을 MCU 에서 쓸 수 있도록 간단하게 만든 통신 프로토콜입니다. 소스는 www.comfile.co.kr 에 공개해 두었습니다.)

MCU 하고는 RS232C 를 사용해서 연결합니다. 작화는 ComfileHMI Editor 라는 프로그램을 이용해서 그리구요. 이 프로그램은 무료입니다.



이렇게 간단해집니다.



최종소비자는 내부에 어떻게 구성되어 있는지 관심없죠? 기능만 나오면 되기 때문에, 엔지니어는 빠르게 원하는 기능을 구현하면 됩니다.

시간이 돈이죠. 개발시간을 단축 한다는건 인건비를 줄이는 것을 뜻합니다. MCU 엔지니어는 코딩의 양이 줄어들고, 회로가 간단해져서, 빠른시간안에 개발을 마칠 수 있을 것입니다. 나중에 그래픽 화면 수정요구가 온다면, MCU 코드를 건드릴 필요없이 HMI 작화 프로그램 상에서 화면을 수정하고 HMI 에 다운로드하는 것만으로도 끝낼 수 있을 것입니다. 유지보수가 한결 간편해지겠지요.

HMI + MCU 솔루션을 알고 있는 당신이라면, 이런 시장의 요구에 좀더 빠르게 대응하고, 유지보수도 원만히 해결해줄 수 있지 않을까요?

제 7 장. 컴파일제품 소개

10만원짜리 산업용 컴퓨터 큐블록

인터넷 서핑하다가 저희 큐블록을 소개한 글이 있어서 퍼왔습니다.
출처 (http://snowxtal.egloos.com/viewer/338425)

연구개발이나 생산기술 쪽 일을 하다 보면 당연히 컴퓨터를 써야 할 일이 많다. 실시간 데이타를 수집한다거나, 데이타를 기반으로 기기를 작동시키거나 하는 일은 역시 사람보다는 컴퓨터가 더 잘 한다. 근데 컴퓨터의 종류는 PC 만 있는 것이 아니다. 단순한 시계, 계산기 등도 일종의 컴퓨터라고 볼 수 있고, 이보다 좀 더 복잡한 것부터 PC 에 이르기까지 그 종류는 무수히 많다. 요즘 유비쿼터스 어쩌고 하는 얘기는 그만큼 컴퓨터가 보편화되어 대부분의 전자기기가 마이크로 프로세서를 내장하고 있다는 사실을 단적으로 표현한 것이다.

이 글에서는 생산/기술 업무상 제어용으로 사용할 수 있는 컴퓨터의 종류와 나의 개인적인 경험을 나열해 본다.

1. PLC (programmable logic controller)

왕 옛날 애니악이 탄생했을 때, 프로그래밍이란 지금처럼 통합개발 환경 화면 앞에서 타이핑하는 것이 아니고 케이블 더미를 여기 저기 끌고 다니며 특정 위치에 꽂아서 데이타 처리순서와 방법을 이 무식한 컴퓨터에 알려줘야 했다. 그런데 이런 방법은 지금도 사용하고 있는데 바로 릴레이 로직이다.

생산기기들은 버튼을 누르면 몇 번 기기를 켜고, 다음에는 무슨 일을 하고, 어떤 센서가 켜지면 중지한다는 식의 단순한 논리를 가지고 있다. 대부분의 전기기술자들에게는 이런 간단한 로직을 구현하기 위해 컴퓨터 앞에 앉아 C 언어 프로그래밍하는 것보다는 릴레이/타이머/카운터 등을 연결하여 배선하는 것이 더 쉽다. 실제 생산기기의 컨트롤 패널을 열어보면 반드시 전원차단 장치와 한 두 개의 릴레이/타이머/카운터 등을 찾을 수 있다.

간단한 로직은 여전히 이런 개별소자를 이용하는 편이 간단하지만, 로직이 복잡해 질수록 사용해야 할 개별소자와 배선의 수는 급속히 증가하게 된다. 입력이 복잡해지고, 배선을 바꿔야 할 경우가 잦아지면 슬슬 이런 배선작업을 간단히 할 수 있는 방법이 없을까 고민하게 된다. 근데 복잡한 릴레이 로직회로를 자세히 살펴보면 입출력에 사용하는 릴레이의 개수는 사실 몇 개 안 된다. 논리를 구현하기 위해 사용되는 소자들과 배선만 늘어나는 것이다. 그래서 고안된 것이 PLC 이다.

PLC 는 물리적인 개별소자를 가상소자로 대체하고, 이들 간의 배선을 소프트웨어적으로 가능하게 한 것이다. 대부분의 상용 PLC 는 수~수십개에 이르는 입출력용 접점과 더불어 내부에 수백~수천개에 이르는 가상소자를 갖고 있다. 사용자는 외부신호, 출력전원 등만을 물리적으로 입출력 접점에

배선하면 되고, 접점 및 내부소자 간의 연결은 레더(ladder)다이어그램이라고 불리는 도면을 PC 에서 그려 다운로드하기만 하면 된다. 그러고 나면 PLC 는 수밀리초(ms) 간격으로 입력접점을 읽어들여, 레더 다이어그램에 그려진 대로 가상소자의 on-off 를 에뮬레이션한 다음, 출력접점으로 결과를 내보낸다. 그러면, 실제 물리적으로 구성한 회로와 아무런 차이가 없는 동작을 하는 것이다.

요즘 들어서는 워낙 마이크로 프로세서와 메모리 가격이 싸져서 개별소자를 여러 개 사는 것보다 저가 PLC 를 하나 사는 것이 더 싸게 먹히기도 한다. 그런 목적으로 사용할 수 있는 PLC 중 하나가 컴파일 사의 CUBLOC 시리즈다. 배선에 들어가는 시간과 소위 '인건비'를 고려하면 10 만원도 채 안 되는 가격으로 수천개의 내부릴레이, 타이머, 카운터, 클럭, ADC 등 하드웨어뿐 아니라 모든 필요한 소프트웨어를 제공하는 이런 제품을 놔두고, 땀 흘리며 좁은 공간에 개별소자를 연결하려는 노력이 미련스럽게 느껴질 수도 있다. 사실 그래서 요즘은 PLC 없이 개별소자만으로 구성한 회로를 구경하기가 점점 어려워지고 있다.

나 같은 비전문가가 간단한 수준의 자동화 동작을 구현하기 위해 이중화 처리가 가능한 수백만원짜리 PLC 를 살 이유는 거의 없다. 지금까지 내가 PLC 를 이용해 만든 것은 순수공급-교반-침강대기-배수의 싸이클을 반복하여 분말을 세정하는 장치라던가, A 탱크 ->B 탱크의 이동이 끝나면 B 탱크->A 탱크로 이동시키며 필터처리를 하는 장치 등 4~5 개의 입력(센서/버튼) 및 1~2 개의 출력(교반기, 펌프)을 갖는 아주 단순한 것들이었다. 그러나, 이런 작업들은 시간이 무척 오래 걸리므로 작업자가 옆에 앉아 각 싸이클을 조작하기에는 너무도 비효율적이다. 이런 경우 겨우 몇 만원짜리 PLC 가 이런 노가다를 대신해 준다면 얼마나 고마운 일인가.

2. PC

나는 지금껏 PC 보다 복잡한 컴퓨터를 사용해 본 적이 없다. 엄청나게 많은 사람들이 PC 를 갖고 엄청나게 다양한 작업을 하고 있다. 물론 PC 를 이용해 자동제어도 할 수 있다. PC 를 이용한 자동제어는 그 자체가 엄청난 경우의 수와 다양성을 갖고 있다. 간단히 예를 들어 보면:

- 입출력 보드 및 전용 소프트웨어를 쓰는 방법
- MatLab, LabVIEW 등 범용 제어 소프트웨어를 이용한 방법
- C, BASIC 등 범용언어를 이용해 프로그래밍하는 방법
- 기타 꼼수를 쓰는 방법(엑셀+RS232 등)

PC 를 쓰는 자동화는 이 글에서 소개하는 다른 방법과 비교할 때 몇 가지 장단점이 있다.

-장점: 데이타 저장/이동이 편리하다. 친숙하다. 언제든지(심지어 자동제어 작업중에도) 다른 용도로 사용이 가능하다.

-단점: 비싸다. 쓸데없는 기능이 너무 많다. 현장에 설치해 놓으면 이상하게도 jpg/mp3/avi 등의 확장자를 가진 화일들이 마구 늘어나며 시스템 장애를 발생시킨다...

개별방법에 대해서는 여기서 구체적으로 언급하지 않겠다. 사실 개인적으로 멀티태스킹 윈도우즈 환경은 실시간 하드웨어 제어가 필요한 자동제어 분야에는 그다지 적합하지 않다고 생각하며,

실제로 RS232 를 사용하는 이외에는 직접 프로그래밍해가며 자동화 프로그램 만들어 본 적이 없다. 디바이스 드라이버 처리방법도 복잡할 뿐만 아니라, 고가의 컴파일러 및 제어 소프트웨어를 사야 하는 등 오버헤드가 너무 크기 때문이다. 그래도 반드시 PC 로 자동화를 해야겠다고 생각하는 분은 내쇼날 인스트루먼트의 LabVIEW 및 관련 솔루션이 도움이 될 지도 모르겠다. 내가 직접 한 것은 아니지만 개발을 의뢰하여 소형 공정자동화 프로젝트를 단기간에 성공적으로 마친 경험이 있다.

3. CUBLOC

오래전 대학원 때 도스환경에서 PC 프린터 포트에서 선을 따내어 파워 트랜지스터 붙여 스테핑 모터를 돌려가며 분광실험을 했던 이후 윈도우즈 환경으로 넘어가면서부터 나는 하드웨어 제어 프로그래밍을 오랫동안 잊고 지냈다. 개별 프로그램의 하드웨어 독점을 금지하는 윈도우즈 환경에서 프로그램 짜는 것도 낯설고 힘들었거니와, 회사생활하면서 그렇게까지 해가며 실험할 일은 별로 없었기 때문이다. 필요하면 좀 비싸더라도 그냥 사서 쓰는 편이 편리하고 빠르니까..

그러나 살다보면 특히 프로그래밍 세계에서는 자신이 한번 했던 것, 그리고 잘 했던 것은 자꾸만 반복하게 되는 경향이 있다. 대학원에서 스테핑 모터 돌린 이후 잊고 지낸 지 10 년, 어느 날 나는 회사 생산공정에서 온종일 앉아서 밸브만 열었다 닫았다 하는 오퍼레이터들을 보게 되었다. 페이스트 제품을 통에 담는 포장 공정이었는데 점도가 높고, 2000g 한 통의 허용오차가 +/- 0.2g 밖에 되지 않아 아직까지 수동작업을 하고 있었다. 이를 자동화하기 위한 프로젝트가 영국공장에서 진행되고 있었는데 결과가 그다지 좋지 않은 상태였다. 나는 스테핑 모터를 이용한 제어를 떠올렸고, 별도의 프로젝트를 제안하여 착수하게 되었다.

이 과정에서 찾은 것이 CUBLOC 이었다. 내가 필요한 것은 RS232 통신, 실수연산, 스테핑모터 제어를 위한 고속 디지탈 출력, 버튼 등 디지털 입력 등의 하드웨어 기능과 이를 제어하기 위한 고급언어였다. 이런 종류의 제어는 PLC 로는 도저히 불가능한 것이었고 어떤 종류가 됐건 마이크로프로세서를 쓸 수 밖에 없었다. 이 때 이 모든 기능뿐만 아니라 아예 전원 /릴레이 /스테핑모터 제어함수까지 더해서 단돈 12 만원인 CUBLOC 보드는 나에게는 거의 광명이었다. 개발기간은 생각보다 엄청 단축되었고, 제어장치는 곧바로 적용에 들어갔다.

CUBLOC 은 웬만한 자동화에 필요한 하드웨어를 거의 모두 갖추고 있다. 앞서 말한 기능 외에 카운터, PWM 제어, 외부 인터럽트, 입출력 릴레이, 아날로그-디지탈 변환, 디지탈-아날로그 변환 등의 하드웨어 환경과 레더와 BASIC 을 동시에 사용할 수 있는 개발환경을 제공한다. 그 외에도 풍부한 주변기기들도 지원하는데 LCD, 키패드, 리모콘, 이더넷 접속, SD 카드 데이터 입출력 등도 가능하다. 한 마디로 자동제어에 있어서는 All-in-one 솔루션이다. 이런 기기가 10 년전에 있었더라면 나의 대학원 생활이 좀 덜 고달팠을 거라는 생각이 든다.

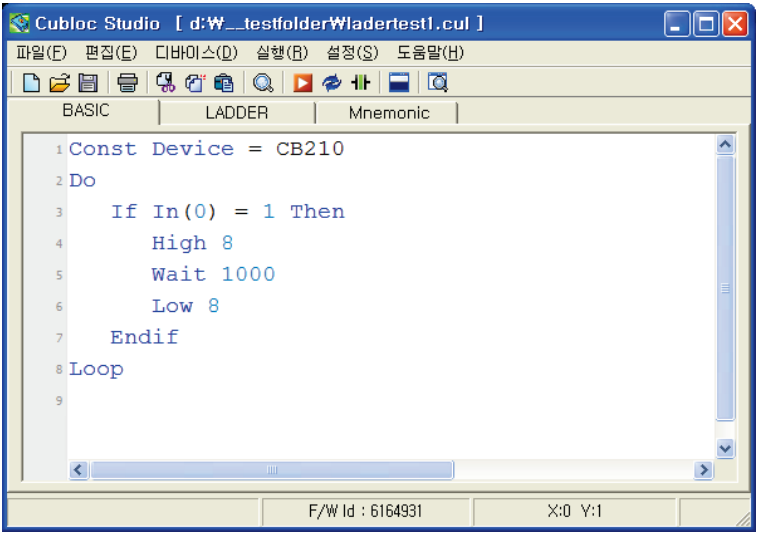
어쨌건 크건 작건 PC 없이 동작하는 스탠드-얼론형 자동화를 고려하는 이에게는 꼭 추천해주고 싶은 제품이고, 아직까지는 유사품도 찾지 못 했다. 자세한 것은 www.comfile.co.kr 을 참조하기 바란다.

레더로직의 가장 큰 장점은 멀티테스킹

큐블록이 다른 콘트롤러(MCU, 모아콘, 아두이노)등과 가장 차별화 된 부분은 바로 레더로직을 지원하고 있다는 것인데, 레더로직은 "동시 다발적인 입력에 즉각 반응한다" 라는 장점을 가지고 있습니다.

이것을 BASIC 이나 여타 다른 언어로 구현하라면 상당한 테크닉 (타임 인터럽트 활용)가 필요한데, 레더 로직에서는 간단하게 실현 가능합니다.

P0 이 입력되면 P8 이 1 초동안 ON 을 유지하다가 꺼지는 동작을 BASIC 으로 구현해 보겠습니다.

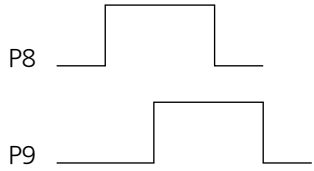


동작도 잘 됩니다. 여기에 P1 번 입력이 들어오면 P9 를 1 초동안 유지시켜주는 동작을 추가해보세요.

```
Const Device = CB210
Do
 If In(0) = 1 Then
  High 8
  Wait 1000
  Low 8
 Endif

 If In(1) = 1 Then
  High 9
  Wait 1000
  Low 9
 Endif
Loop
```

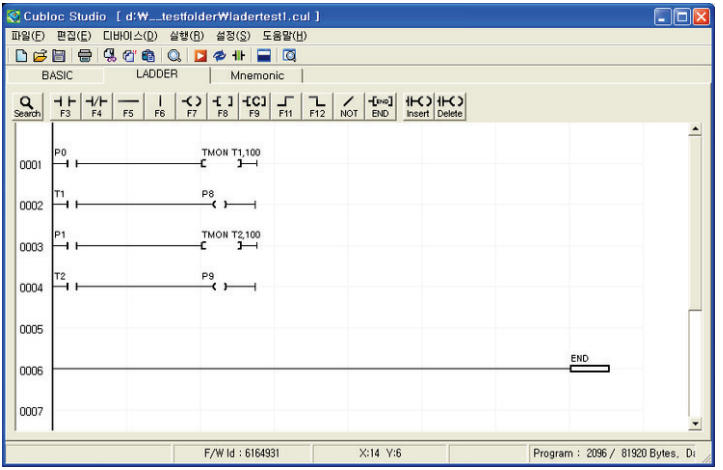
이렇게 하면 동작될 거 같지만, 이건 반쪽짜리 프로그램입니다. P8 이 켜있는 동안 P1 입력을 받을 수 없으니까요.



제대로 동작하려면, P0, P1 입력이 들어오는 즉시 P8, P9 가 각각 1 초씩 유지되어 야 합니다. 서로 간섭받지 말고요.

이제 이걸 BASIC 으로 구현해보세요. 선뜻 아이디어가 떠오르지 않을 것입니다. 이것은 전문가들도 어려워하는 시분할(?)기법이 필요하기 때문입니다.

그런데, 이런 입력처리 상황은 자동화 현장에서 흔히 생기는 상황입니다. P0, P1 에 각각 센서가 연결되었다고 가정해 봤을 때, 센서가 언제 동작할지는 아무도 모릅니다. 센서가 입력되는 즉시, 그에 상응한 출력이 바로 나와주어야, 제대로 기계가 동작할 것입니다. 이러한 입력처리는 레더에서는 쉽게 구현할 수 있습니다.




```
Const Device = CB210
Usepin 0,In
Usepin 1,In
Usepin 8,Out
Usepin 9,Out
Set Ladder On
Do
Loop
```

베이직 쪽에는 레더 실행을 위한 입출력설정 정도만 썼습니다. 이렇게 하고 실행해보니, P0, P1 이 들어오면 각각 P8, P9 를 1 초동안 켜줍니다. 물론 누르는 즉시 반응하고, 다른 쪽의 실행여부와 상관없이 독립적으로 실행됩니다.

바로 이 점이 레더로직이 가지고 있는 최대 장점입니다.

"레더로직은 그 자체가 멀티테스킹 동작을 수행하는 그림 형태의 언어이다."

라고 보시면 됩니다.

모아콘 : 산업용 C프로그래머블 콘트롤러

사실 모아콘은 큐블록의 다음 버전을 생각하는 과정에서 나온 제품입니다.

큐블록을 사용하시는 분들에게 물어보니, 대략 3 가지를 개선해주었으면 하는 것으로 의견이 종합되었습니다.

1. 속도
2. PCB 를 제작해야 하는 불편함
3. I/O 의 다양성 부족

우선 속도를 올리기 위해서 32 비트 ARM 프로세서를 채택했고, PCB 를 제작 해야하는 불편함을 없애기 위해서 모듈형으로 설계했습니다.

I/O 다양성을 충족시키기 위해, A/D, D/A , 온도, 이더넷, 모터제어 등의 모듈을 사전 제작해놓고, 유저들은 단지 취사 선택하여 사용할 수 있도록 한 것입니다.

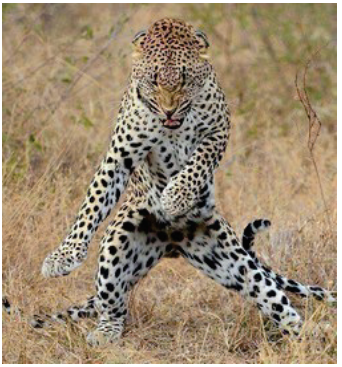
이렇게 하다 보니, 큐블록의 BASIC 언어로는 한계성이 있었습니다. 전세계 프로그래머들중 가장 많은 수가 애용하고 있는 언어가 C 언어라고 하니, C 언어로 설계할 수 있도록 한 것이 바로 "모아콘" 입니다.

큐블록의 다음 버전으로 출발한 프로젝트가 "모아콘"이라는 새로운 제품군으로 탄생된 것입니다. 큐블록은 큐블록 나름의 개성을 갖고 있으니, 계속 발전시켜 나가겠습니다.

모아콘은 "큐블록"으로 부족함을 느끼는 고급사용자들에게 적극 추천하는 콘트롤러입니다.

ARM 과 모아콘

32 비트 MCU 를 잘 다루시고, PCB 하나정도는 뚝딱 하고 디자인 하시는 분들은 "모아콘"을 쓰지 않을 듯 싶습니다. 이 분들은 전기/전자라는 무림의 세계에서 "장풍을 쓰는 고수"이십니다. 디자인 노우하우가 풍부하고, 노이즈 차단, EMI 차폐정도는 늘 하던 일이며, C 프로그래밍 정도는 누워서 떡 먹듯이 쉽게 하실 줄 아는 분들입니다.



(세랭게티의 무림고수랍니다. ㅋ)

저희 회사의 모토는 "초보자도 쓰기 편리한 임베디드 콘트롤러"를 세상에 내놓는 일입니다.

사실 ARM 프로세서 하나를 공부하는 데에도, 힘겨워 하시는 분들이 많습니다. 데이터북은 왜 이렇게 두꺼울까요! 그리고 C 언어를 안다고 바로 프로그램을 짤 수 있는 걸까요?

32 비트 ARM 정도 되는 MCU 라면, 0 번지부터 프로그램을 짜는 방식으로 접근하는 분은 없을 겁니다. RTOS 를 사용하거나, 그에 준하는 라이브러리가 필요합니다. ARM 프로세서가 너무 좋다보니, 페리퍼럴도 상당히 많습니다. 그 많은 페리퍼럴들의 레지스터맵만 봐도 골치가 아픕니다.

초심자들은 이런 난관들을 다 극복하고, LED 하나 깜박거리게 하는데까지 도달하는데도 상당한 시일이 걸리는게 현실입니다.
모아콘은 제품을 사서, 그 즉시 컴퓨터와 USB 케이블로 연결한 뒤, 샘플프로그램을 입력해서 다운로드 하면 동작됩니다. 라이브러리는 사전에 준비되어 있으며, 설명도 잘 되어 있습니다.

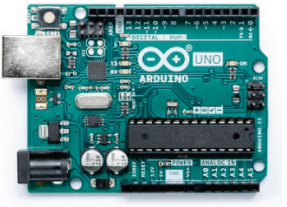
별도의 C 컴파일러를 구입하거나, JTAG 에뮬레이터를 구입하실 필요도 없습니다. USB 케이블 하나로 개발이 가능하죠. "32 비트 ARM 을 다루기가 뭐 이리 쉽나?" 생각하실 정도로 쉬운 콘트롤러가 바로 "모아콘"입니다.

회로를 설계하실 필요도 없습니다. 대부분의 I/O 가 모듈로 제품화 되어 있으니, 필요한 것을 모아 모아서 꼽으면 그만입니다.

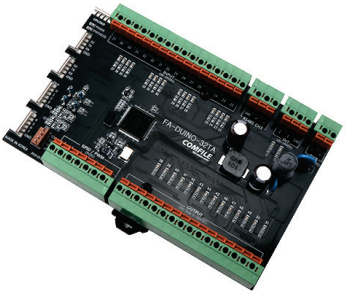
초보자들도 쉽게 쓸 수 있는 ARM 기반 산업용 콘트롤러가 바로 모아콘입니다.

산업용 아두이노 FA-DUINO

아두이노가 뭔지는 다들 아시죠? 교육용, 입문용으로는 아주 훌륭한 제어용 보드입니다.



하지만 산업용으로 쓰기에는 뭔가 부족한 점이 많아 보이지요? 그래서 산업용으로 쓸 수 있도록 만든 아두이노 FA-DUINO 를 출시하게 되었습니다.

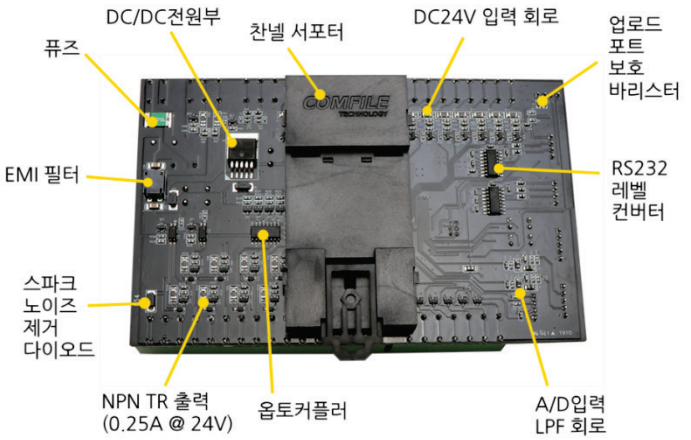


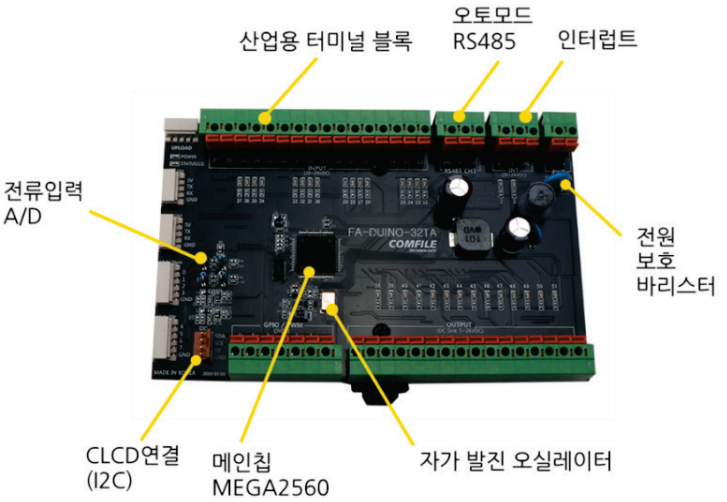
크기도 기존 아두이노 보다는 크지만, 기능면에서도 많이 추가가 되었는데요. 중점적으로 전원 입력부와 I/O 입출력부를 산업용에 적합하도록 보강을 하였습니다.

꼽았다 뺐다 할 수 있는 산업용 터미널블록을 채택했습니다. 기존 아두이노처럼 잡아 땡기면 쉽게 빠지는 구조가 아닙니다.

아날로그 입력과 PWM 출력, 인터럽트 입력, 그리고 I2C (CLCD 디스플레이용으로 사용가능)도 가지고 있어서 여러가지 목적에 폭넓게 이용할 수 있도록 하였습니다.

다음은 FA-DUINO 32TA 제품의 구성요소를 설명한 그림입니다.





제 8 장. 좋은 코딩 습관

구조적 프로그래밍

좋은 프로그램이란 무엇일까요? 물론 문제없이 동작되야 하는건 기본이구요. 그리고 에러도 적고, 나중에 봐도 쉽게 이해할 수 있는 프로그램이 좋은 프로그램이겠죠. 결론부터 말씀드리자면, "구조적 프로그래밍"을 해야, 좋은 프로그램을 만들 수 있습니다.

"구조적 프로그래밍"이란 무엇이냐?

말그대로 구조적으로 프로그램 소스를 작성하는 것을 말합니다. 그러려면 몇가지 중요한 룰을 지켜주어야 합니다.

첫번째.
보기좋은 떡이 먹기도 좋은 법입니다. 들여쓰기를 지키고, 주석을 많이 달아야 합니다. 작명에도 신경 써야 합니다. 변수명은 변수의 역할을 알 수 있는 이름으로 작명해야 하고, 함수명도 마찬가지입니다.

abc123 과 같은 무의미한 문자보다는 Delay_Rtn 과 같은 이름만으로 의도를 알 수 있는 이름을 사용해야, 나중에 독해하기 편리합니다.

Good

```
For i = 0 to 5
   If mAdin(i) = mMoveDownOk(i) Then
     Out mDNRelay(i),0
   Else
     Out mDnRelay(i),1
   Endif
Next
```

Bad

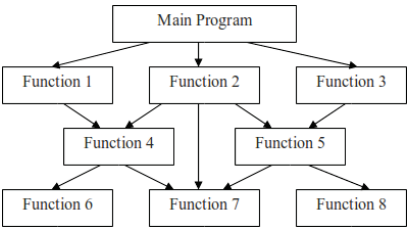
```
For i = 0 to 5
If abc(i) = g5(i) Then
Out 13,0
Else
Out 13,1
Endif
Next
```

같은 숫자가 소스에서 여러번 등장한다면 상수정의를 이용하세요. 숫자가 어떤 의미로 사용되는지 한눈에 알 수 있고, 나중에 한꺼번에 변경할 때도 용이합니다.

```
Const TimerMaxValue = 100
```

두번째.

여러 개의 모듈(함수)을 만들고, 그 모듈들이 모여서 하나의 프로그램이 구성되도록 만들어야 합니다.



세번째.

모듈의 독립성이 보장되어야 합니다. 즉, 모듈안에서 변수도 정의하고 (지역변수), 모듈안에서 모든 처리를 끝낸 뒤, 그 결과를 도출해내야 합니다. 이렇게 작성된 모듈은 나중에 다른 프로그램을 짤 때, 그대로 가져다 쓸 수도 있습니다.

그리고, 가급적 전역변수를 사용하지 마세요. 지역변수는 함수실행시 생성되었다가 함수 수행이 끝나면 자동 소멸되지만, 전역변수는 항상 메모리를 차지합니다. 게다가 이 함수, 저 함수 안에서 전역변수 내용을 마구잡이로 고친다면, 예기치 못한 에러가 발생할 수도 있습니다. 함수의 독립성도 떨어집니다.

구조적으로 프로그램을 짜시게 되면, GOTO 명령이 필요 없다는 것을 아시게 될겁니다. GOTO 명령은 사용하지 마십시오. GOSUB 명령도 지역변수를 선언할 수 없기 때문에 사용하면 안됩니다. 구조적 프로그램을 짜는 습관을 처음부터 익히시면, 나중에 두고두고 편합니다.

Blocked 프로그램을 만들지 말자

이번에는 Blocked 프로그램이 무엇인가에 대해서 설명드리겠습니다.

미국 여행을 다니다 보면, 이런 교통표지판을 보게 됩니다. DO NOT BLOCK 이게 뭔소리냐 하면, 도로를 막지 말라는 뜻입니다. 주차를 아무렇게 하거나, 정차를 하더라도 길을 막는 식으로 하지 말라는 것이지요.

프로그램도 "Do not block" 으로 짜야합니다. Blocked 프로그램이란 소스 중간 중간에 흐름을 막는 부분이 산재해있는 프로그램을 말합니다.

예를 들면, 대기하는 상황에서 무한루프로 대기해서는 안됩니다.

```
Do  WHILE In(12) = 1
Loop
```

이렇게 명령을 내리면, 포트 12 가 0 이 될 때까진, 아무것도 안하고 무한 대기합니다. 만약 포트 12 에 문제가 생겨서 0 이 될 수 없다고 하면, 이 프로그램은 더이상 진행을 안하고 여기서 멈춰버리게 되고 말 것입니다. 이런 경우라도, 최대한의 리밋이 있어야합니다.

```
For i = 0 to 100000
    if In(12) = then Exit for
Next
```

For 루프가 10 만번 도는 동안만 기다리게 하는 방식입니다. 충분히 기다렸는데도 반응이 없다면, 그에 대한 처리를 한 뒤, 다음으로 진행하도록 프로그램을 짜야합니다.

MCU 라면 "워치독 타이머"를 이용하는 것도 좋은 방법입니다.

```
CLRWDT
LOOP1:
BTFSC PORTA,0
GOTO LOOP1
```

이렇게 한다면, 루프에 들어가기 전에 와치독을 클리어 하고 들어갑니다. 그리고 포트 A0 이 LOW 가 될 때까지 무한 대기하는 것 같지만, 사실은 와치독 타이머가 돌아가고 있습니다. 루프안에는 클리어 와치독(예:CLRWDT) 명령어가 없으므로, 언젠가는 와치독 타이머가 오버플로우되서 MCU 를 리셋시켜 줍니다.

기억해두세요. 어딘가에서 멈춰있지 않고, 물 흐르듯이 원활하게 흐르는 프로그램이 좋은 프로그램입니다.

단순화부터 하자

큰 프로젝트의 경우 완성된 소스 코드는 3000 행이 넘어가는 경우도 있습니다. 이렇게 큰 프로그램을 작성하려면 웬만한 전문가라도 엄두가 안나게 됩니다.

여러분이 아무리 큰 프로젝트를 맡았다고 하더라도 겁을 먹거나 두려워할 필요가 없습니다. 프로그램이 처리해야될 일을 잘게 쪼개보십시오. 더이상 쪼개지지 않을 때까지 말입니다.

그럼 가장 핵심이 되는 한 덩어리가 떨어져 나오게 됩니다. 단 몇 줄의 소스로 그 부분이 동작되는지 먼저 작성해보세요.

보일러를 예로 들어보겠습니다. 가정용 보일러는 온도입력, 보일러가동, 유저 인터페이스, 물보충 경보, 타이머, 온수조절 등의 다양한 기능을 가진 제품입니다.

이 많은 기능을 언제 만들지? 하고 생각하지 마시고, 일단 핵심부분이 무엇인지를 생각해보세요. 현재 온도를 읽어오는 부분일 것입니다.

그 부분부터 작성을 해보시고, 테스트를 해보세요. 완성이 되었다면, 그 다음단계를 생각해보세요. 이런식으로 살을 덧붙여 나가는 식으로 소스코드를 작성하십시오.

한 단계 살이 덧붙여졌다고 생각되었을때, 지금까지의 기능을 전체적으로 테스트를 해보세요. 이상없이 동작한다구요? 그럼 다음단계로 가시면 됩니다.

제가 말씀드리는 건 STEP BY STEP 의 중요성입니다. 마치 계단을 오르듯이 차근차근 개발을 진행해 나가는 것입니다.

아무리 복잡한 프로젝트라도 여러 개의 단순한 조각들로 구성되어 있고, 최초의 단순한 조각부터 개발해서 살을 붙이고, 서로 맞추어 나가는 식으로 개발을 진행하시면, 어느덧 완성하게 됩니다.

노이즈에 강한 프로그래밍

MCU 에는 여러가지 메모리가 있는데, 쓰임새에 따라 다르게 부릅니다.

- PC (프로그램 카운터) : 프로그램 메모리 중 현재 실행할 번지.
- SP (스택 포인터) : 스택 저장 지점을 가지고 있음.
- 레지스터 : ALU 에서 사용하는 데이터를 가지고 있음.
- I/O (페리퍼럴) 레지스터 : I/O 등 페리퍼럴의 셋팅상태등을 가지고 있음.

다양한 쓰임새가 있지만, 이 모든 것은 그냥 플립플롭으로 구성된 메모리입니다. 노이즈에 의해서 차별대우를 받지 않는 그냥 평범한 메모리인 것입니다. 이중 어떠한 메모리도 노이즈의 공격으로부터 자유롭지 않습니다. 심지어 PC (프로그램 카운터) 조차도 노이즈로 인해 내용이 파괴될 수 있습니다.

I/O 레지스터의 내용이 노이즈로 인해 바뀐다면, 그냥 I/O 상태가 바뀌는 것이지만, PC (프로그램 카운터)의 내용이 바뀐다면, 그 결과는 매우 치명적입니다. 갑자기 알 수 없는 번지로 점프하게 되고 맙니다.

하필 그 번지에 니모닉이 아닌, 데이터가 들어 있다면, 프로그램은 삼천포로 빠지게 됩니다. 데이터는 의도되지 않는 숫자이기 때문에 어떤식으로 실행될지는 아무도 알 수 없습니다. 심지어 MCU 를 설계한 사람 조차도 말입니다.

"노이즈에 의해 PC(프로그램카운터)의 내용이 바뀔 수 있다" 는 가정하에, 프로그램을 짜야합니다. 이런 상황을 피하기 위한 가장 좋은 방법은 와치독 타이머를 활용하는 것입니다. 와치독 타이머를 활용하면 프로그램이 엉뚱한 곳에서 계속 무한루프를 도는 멀펑션 상황에서 탈출한뒤 리셋번지부터 시작하게 해줍니다. 적어도 그곳에는 제대로된 코드가 들어 있으니까요.

와치독 타이머가 오버플로우되면 MCU 는 리셋됩니다. 프로그램 곳곳에서 CLRWDT (클리어 와치독)과 같은 명령을 써서 (MCU 마다 명령어는 다름) 와치독 타이머를 0 으로 만들어주세요. 그러면 프로그램이 정상동작 중일 때 MCU 는 리셋되지 않게

됩니다. 그럼 I/O 레지스터가 잘못되는 것은 어떻게 처리해야 할까요? I/O 레지스터도 노이즈의 공격대상이 될 수 있습니다.

그래서 루프구조로 프로그램을 짜야 합니다. 루프안에서 중요한 I/O 레지스터는 계속 셋팅값을 유지시켜주어야 합니다.

```
DDRA = 0xff;
while (1) {
    input_proc();
    output_proc();
}
```

이런식으로 PORTA 의 입출력상태를 결정하는 명령어를 루프 바깥에 두지 마시고,

```
while (1) {
    DDRA = 0xff;
    input_proc();
    output_proc();
}
```

이런식으로 루프 안에 두십시오. I/O 제어도 마찬가지입니다. 설령 노이즈에 의해서 출력 값이 바뀐다 하더라도, 정상적으로 루프를 한바퀴 더 돌고 나면, 원래 값으로 돌아올 수 있도록 프로그램하십시오.

사실 PLC (프로그래머블 로직 콘트롤러)가 노이즈에 강한 이유가 여기에도 있습니다. PLC 의 레더로직은 루프구조로 되어 있어, 모든 프로그램이 수 마이크로초 간격으로 반복해서 실행합니다. 설령 노이즈에 의해 잘못된 출력결과가 나간다 하더라도, 다음스캔에서 이를 바로잡아줍니다.

여러분이 C 언어나 기타 다른 언어로 프로그램을 짜신다 하더라도, 레더로직처럼 전체적인 상황을 루프를 돌면서 제대로 된 값이 출력되도록 유지시켜주는 방식으로 프로그램을 짠다면, 그것이 바로 노이즈에 강한 프로그램을 짜는 요령이 되겠습니다.

맺음말

현재 저희 회사에는 상당히 많은 기술문의 전화가 오고 있습니다. 주로 제품을 개발하면서 부딪히는 문제해결을 위해 전화주시는 분들입니다.

사실 이 책에 있는 내용들도 고객분들과 통화를 하면서 제가 느꼈던 부분과 추가설명이 필요한 부분을 정리한 것이기도 합니다.

많은 엔지니어 분들이 다양한 문제(?)들과 씨름하면서 고생하고 있는 것을 잘 알고 있습니다. 이 책이 그런 분들에게 조금이나마 도움이 되었으면 하는 바램입니다.