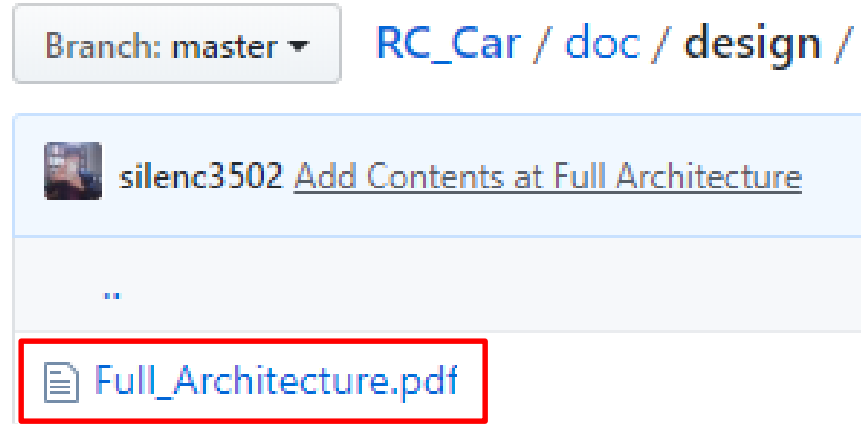


# Xilinx Zynq FPGA, TI DSP, MCU 프로그래밍 및 회로 설계 전문가 과정

강사 – Innova Lee(이상훈)  
[gcccompil3r@gmail.com](mailto:gcccompil3r@gmail.com)

# ***Current Mission Progress***


## 전체 아키텍처에 대한 설계 문서











전체 구성에 대한 아키텍처를 이해하기 위해 만든 문서임  
이 문서를 통해 본인이 뭘 해야 하는지 감을 잡길 바람

## DC-DC 컨버터 개발과 관련한 작업들

Branch: master ▾ RC\_Car / circuit / dcdc /

 silenc3502 Merge pull request #32 from HyunwooParkk/master ...

..

 BUCK CONVERTER .pdf	BUCK CONVERTER
 LC Resonance.pdf	LC Resonance(LC 공진) Circuit
 PSIM PID제어.pdf	PSIM TOOL/PID
 PSIM STUDY 1.pdf	PSIM TOOL STUDY
 PSIM STUDY 2.pdf	PSIM TOOL/PID
 SMPS_PowerElectronics_1.pdf	SMPS Power Electronics
 Switch-Mode Power Supplies.pdf	switch mode power supplies - chapter 2
 prepare.txt	Make DCDC Converter Data Dir


향후 PCB 관련 정보는 PCB 디렉토리에 들어갈 것임

## MCU, DSP, FPGA 관련 문서들











Branch: master ▾ RC_Car / experiment / doc /	
silenc3502 FTDI USB2CAN	
..	
esp8266	esp8266-esp-01 datasheet
AM5728_FTDI_USB_2_CAN.pdf	FTDI USB2CAN
AM5728_WiFi_Lab.pdf	AM5728 Based Wi-Fi Lab
Ardu_Based_ESP8266.pdf	<a href="#">Arduino Based Wi-Fi Module(ESP8266) Test</a>
CAN_Test.pdf	Dedicated Doc for Experiment
Configuring CAN TMS570.txt	Dedicated Doc for Experiment
Cortex_R5_I2C_Howto.pdf	Cortex-R5F I2C Howto
DSP_CAN(using_Serial_communication).pdf	How to use CAN2USB module in DSP
FreeRTOS_Guide.pdf	FreeRTOS manual
HET_PWM.pdf	HET Based PWM
Pmod_CAN_Control_with_Zybo.pdf	Pmod CAN Control with Zybo
RTI_GPIO_OC_Config.pdf	RTI Based Common Emitter Circuit
SPI_Comm.pdf	SPI Communication
SPI_Loopback.pdf	SPI Lookback Exam
stepmotor_test.pdf	How to work step motor in MCU
uart_test.pdf	Dedicated Doc for Experiment

공공기관에서 나오시는 분들을 상대하기 위해 반드시 만들어야 하는 것들임

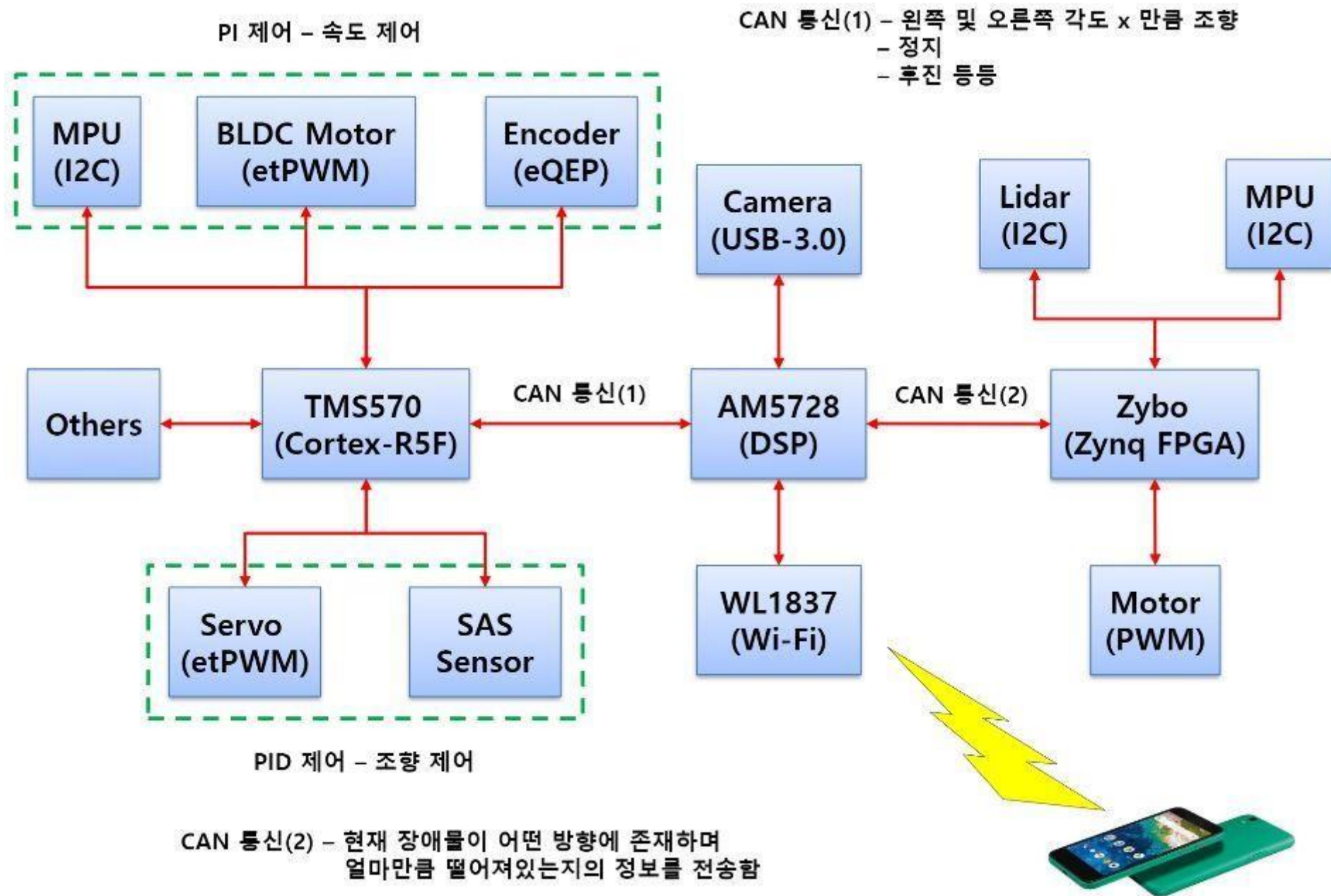
Branch: master ▾ RC\_Car / real\_test /

 silenc3502 NCS - 운영체제 커널분석

..

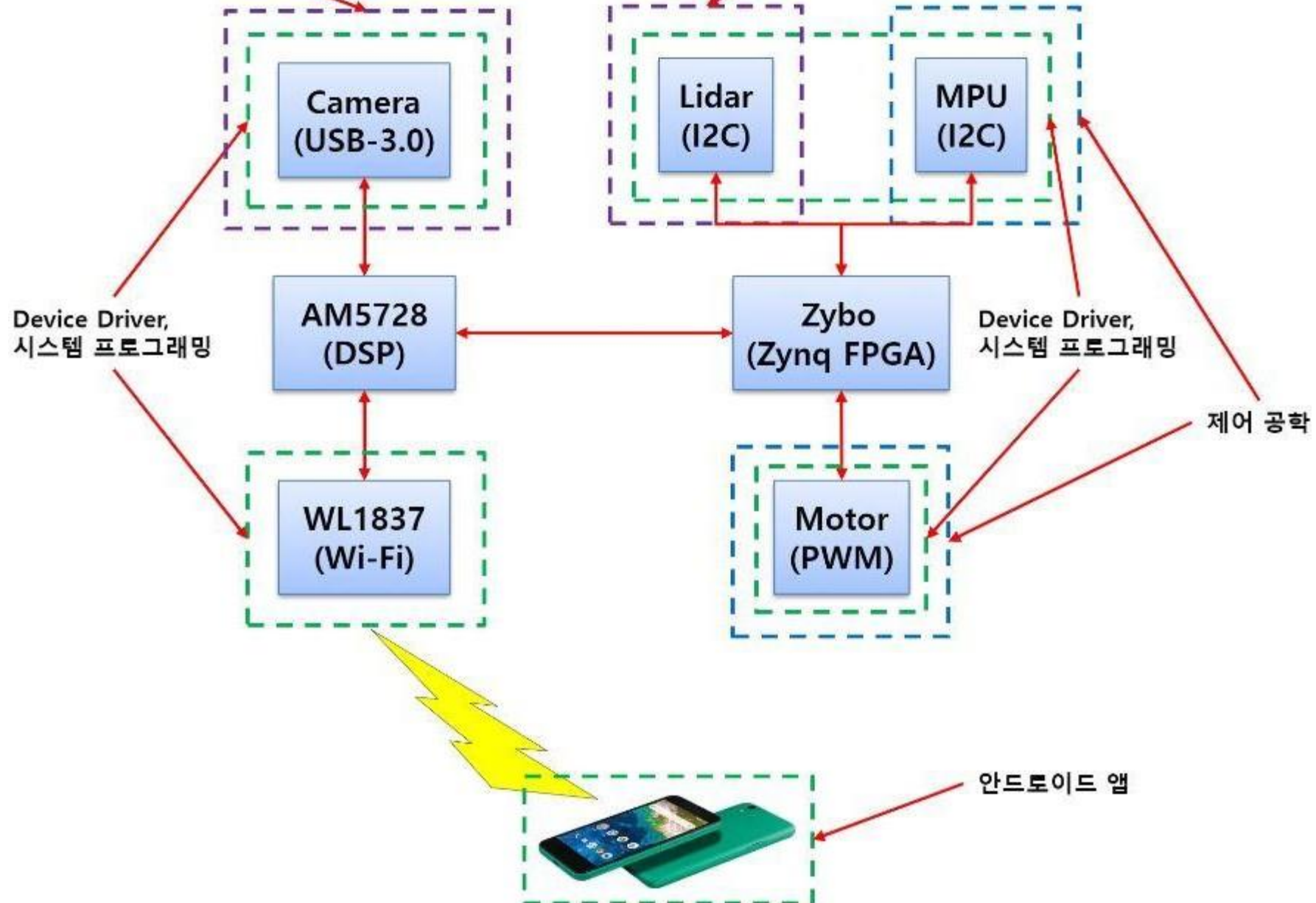
 prepare	adjust folder
 기술문서개발.txt	NCS - 기술문서개발
 오픈플랫폼활용.txt	NCS - 오픈플랫폼활용
 운영체제커널분석.txt	NCS - 운영체제 커널분석
 임베디드시스템테스팅.txt	NCS - 임베디드시스템테스팅
 펌웨어구현.txt	NCS - 펌웨어 구현
 펌웨어구현환경구축.txt	NCS - 펌웨어 구현 환경 구축
 펌웨어분석.txt	NCS - 펌웨어분석
 펌웨어설계.txt	NCS - 펌웨어 설계
 하드웨어분석.txt	NCS - 하드웨어 분석 시험 및 답안지

시험 문제와 답안을 전부 올려놴으니 대충 60 점만 넘기도록!!!  
나중에 몇몇 사람들 선별해서 인터뷰도 존재함!

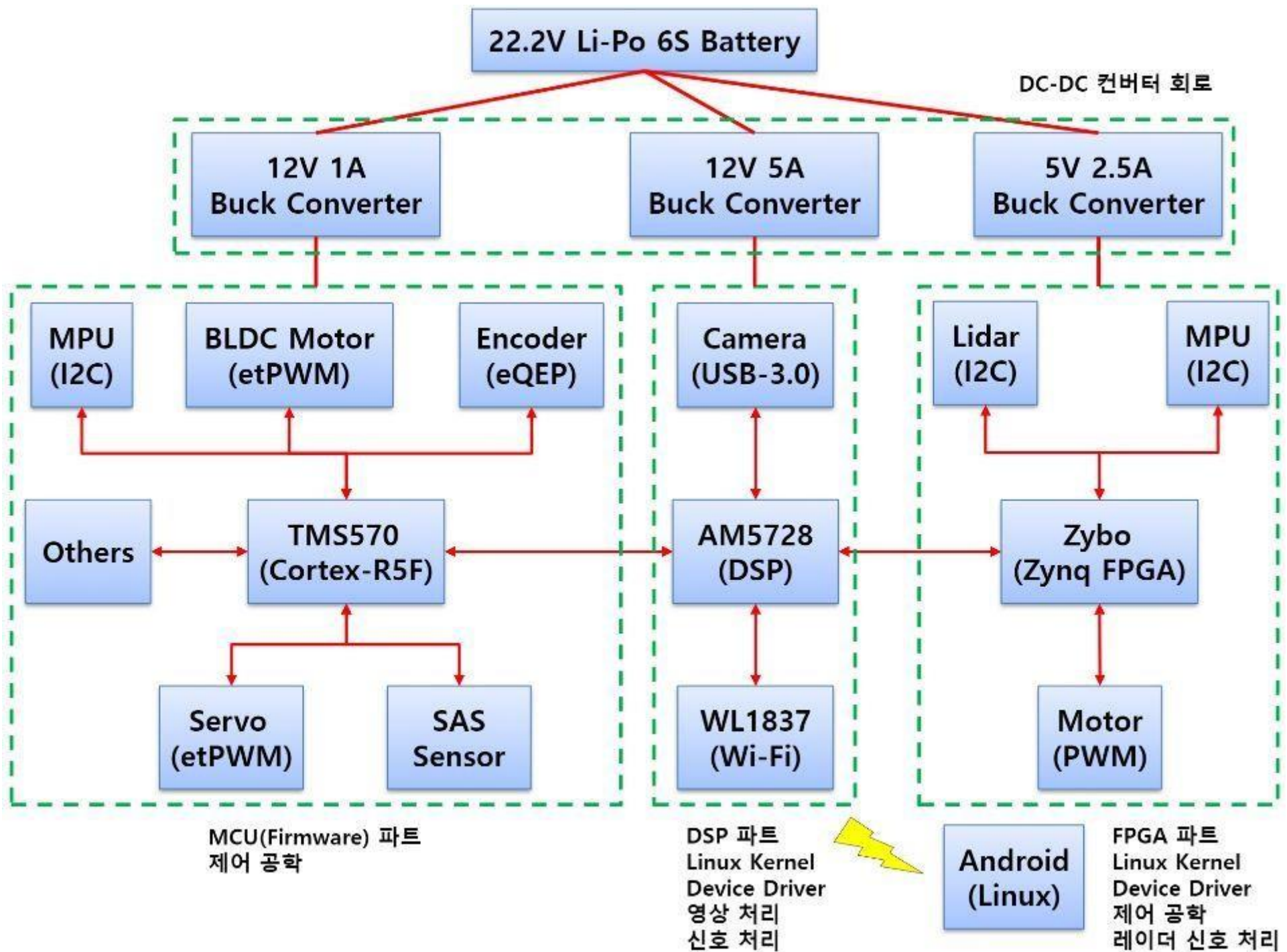


영상 신호 처리

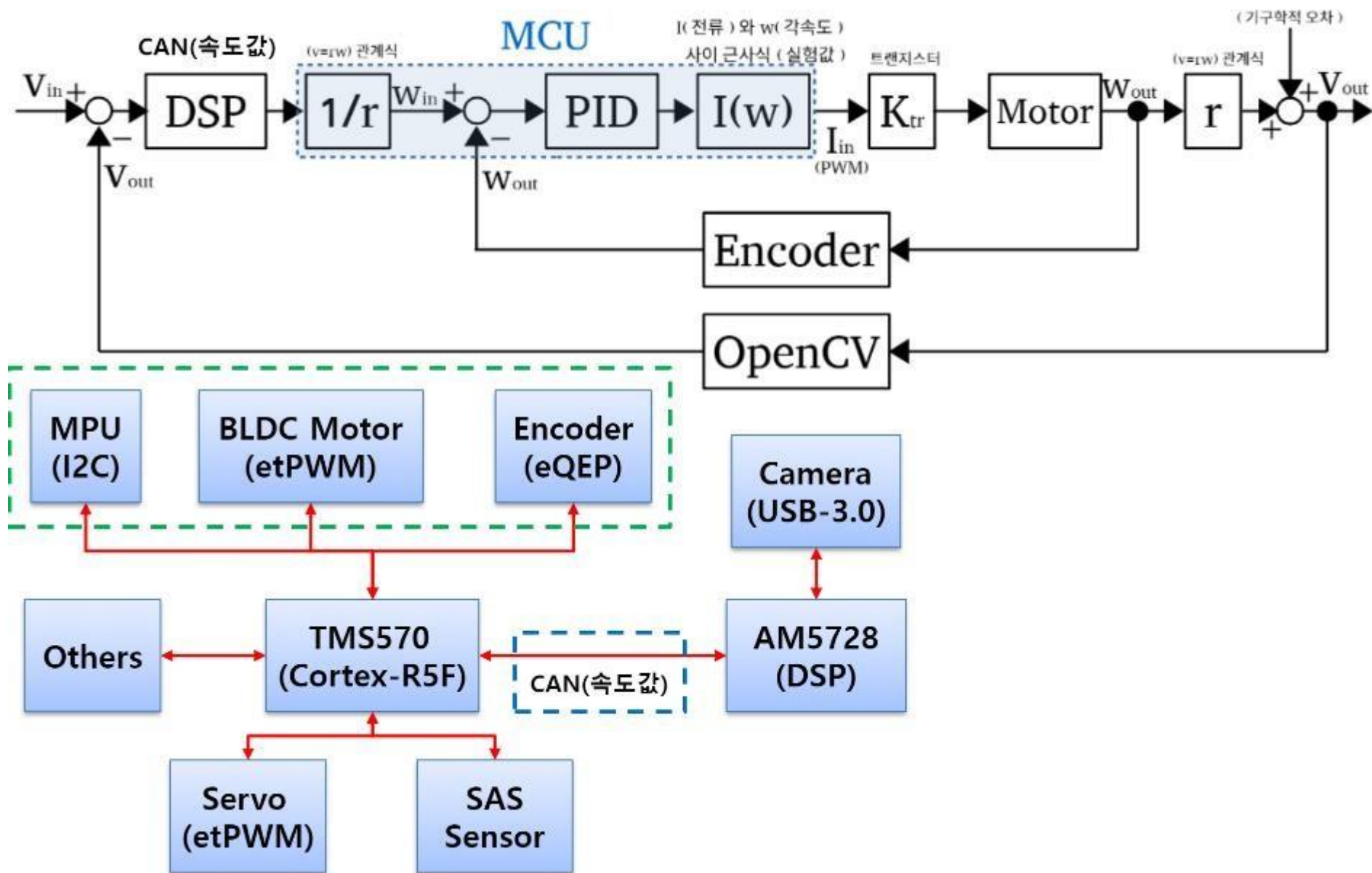
레이더 신호 처리



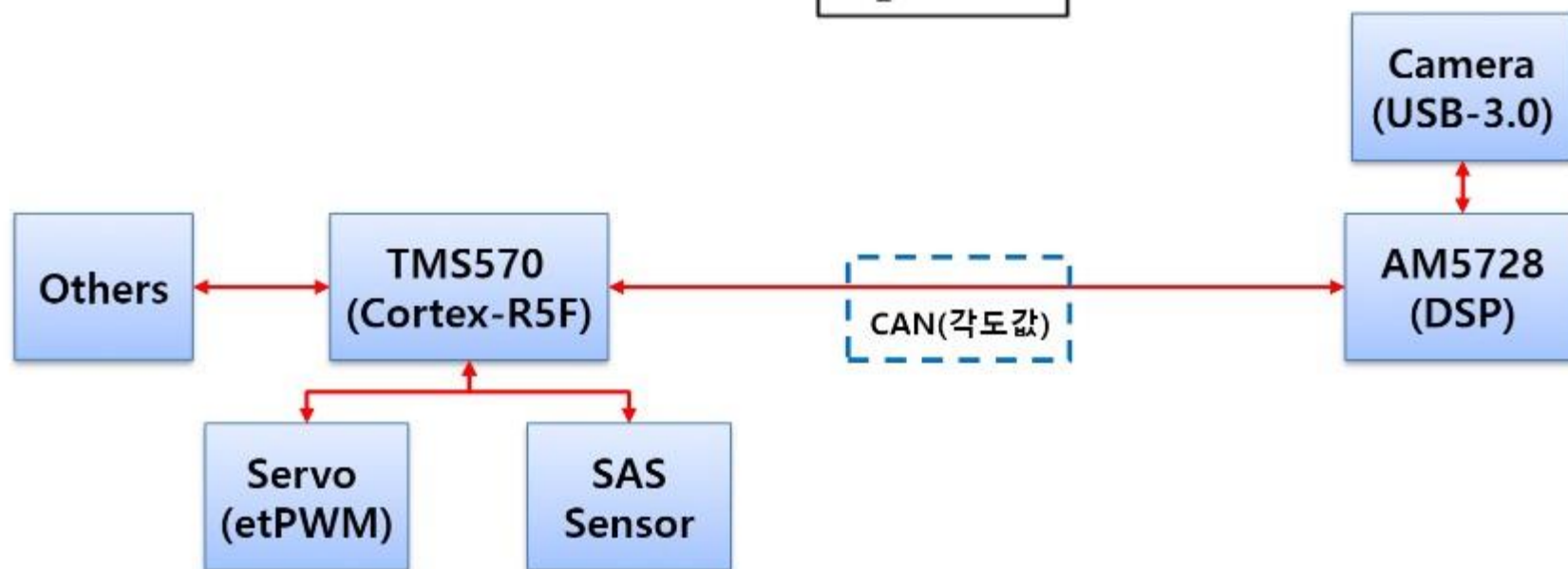
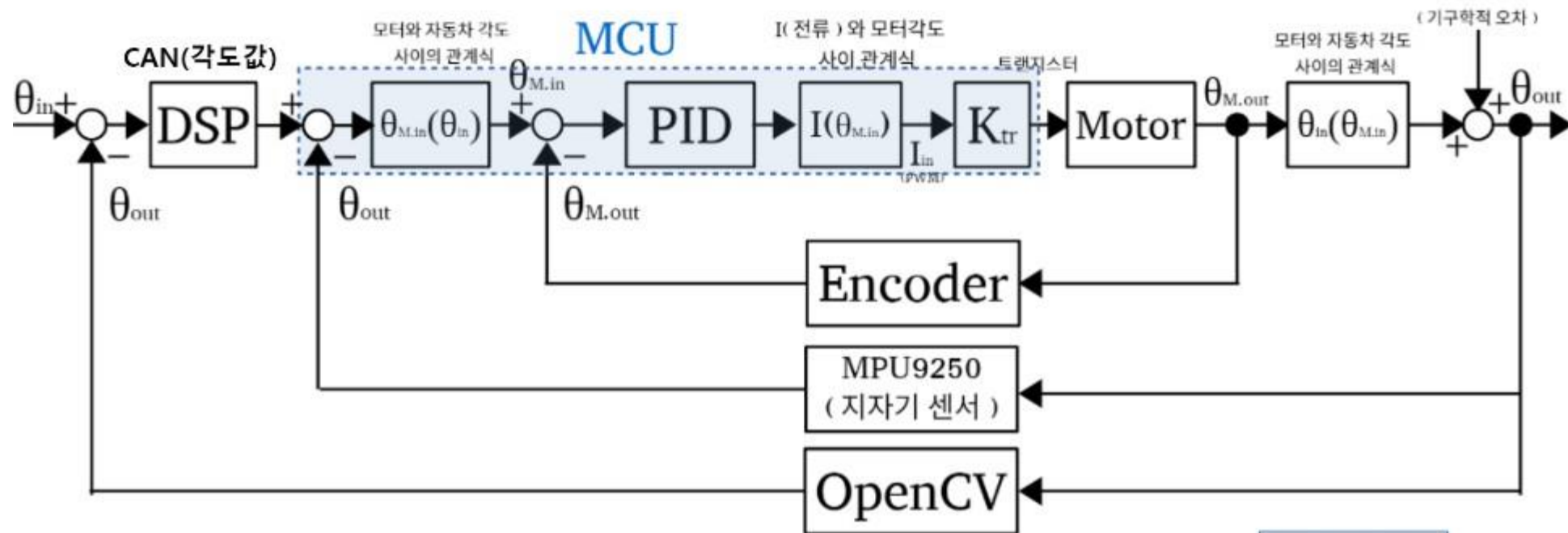




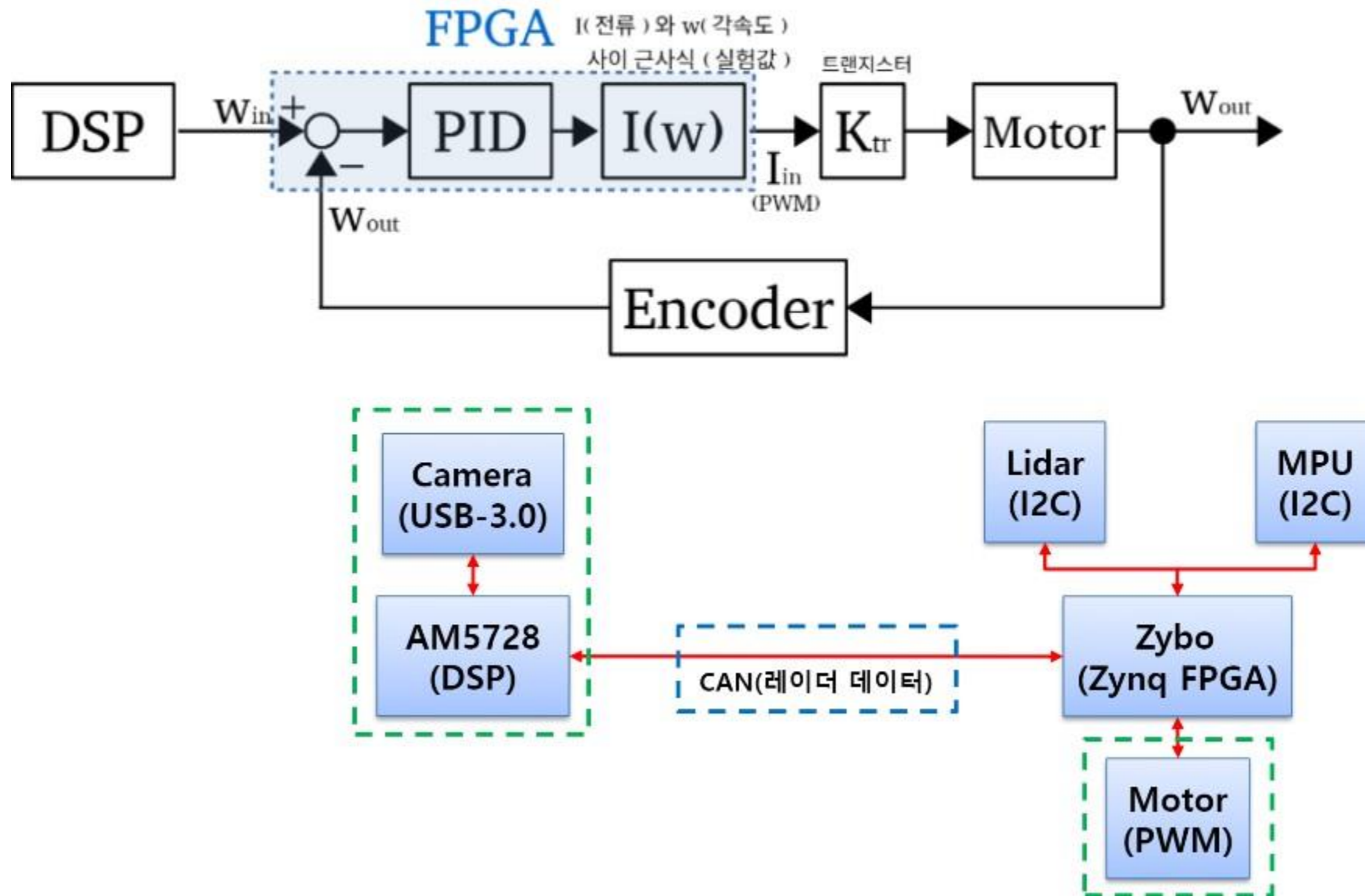
## 자동차 속도 PID 제어



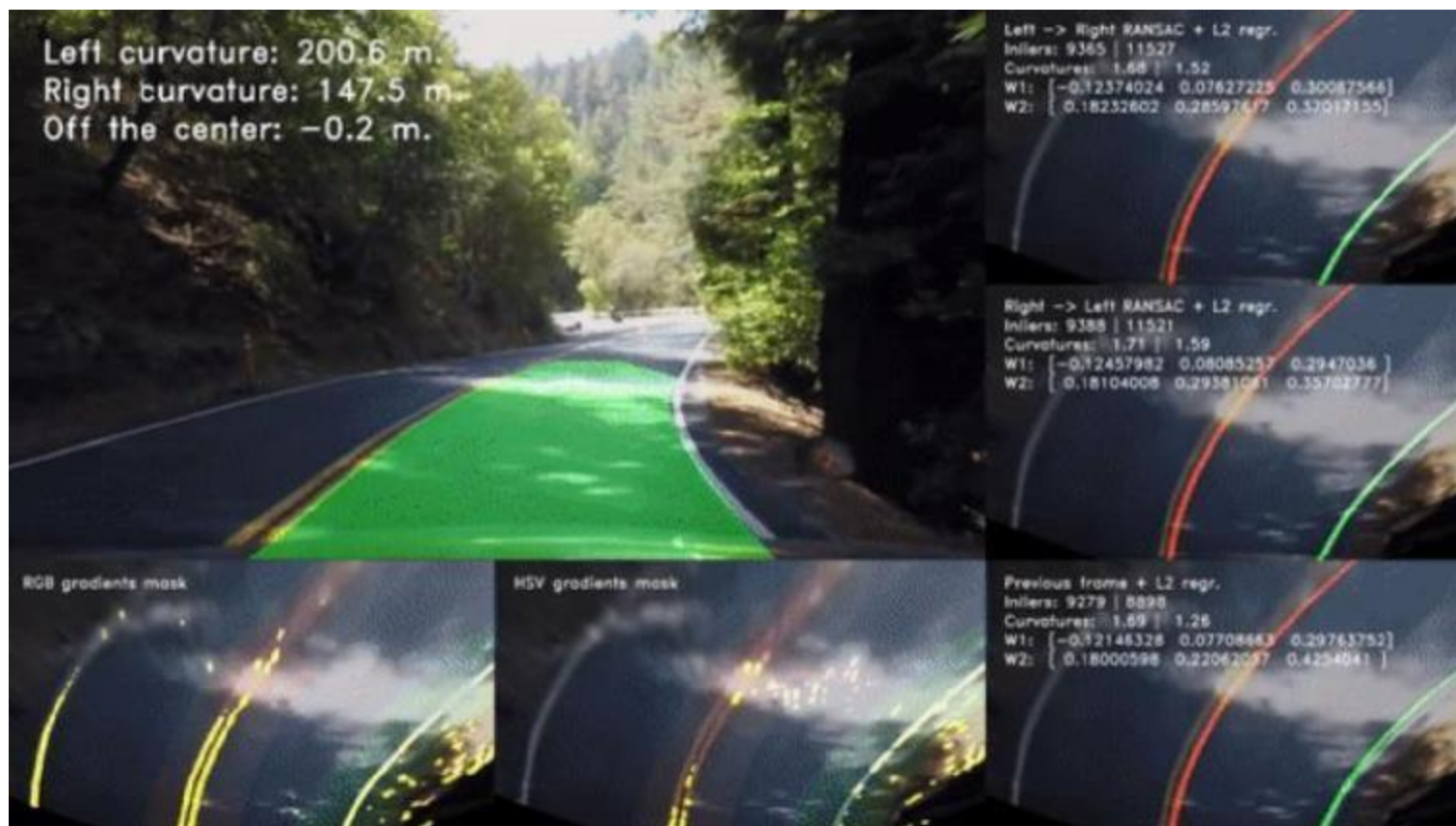
# 자동차 조향 PID 제어



# 라이다 모터 PID 제어









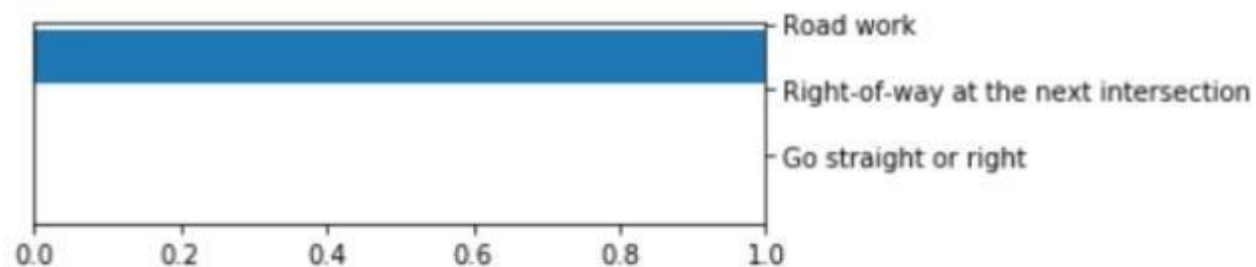
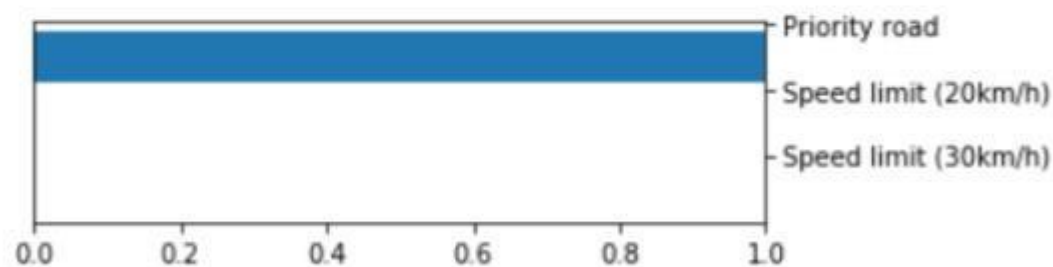
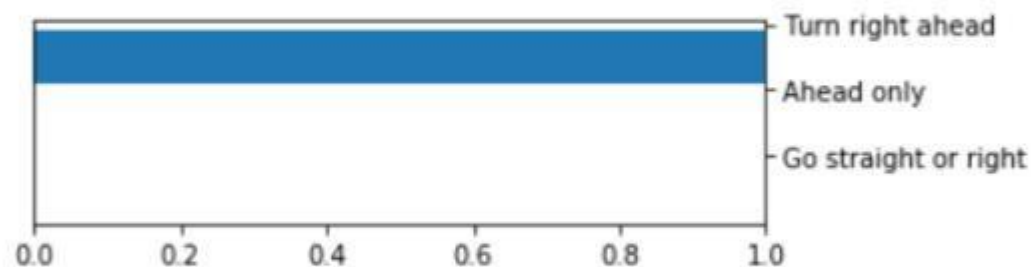
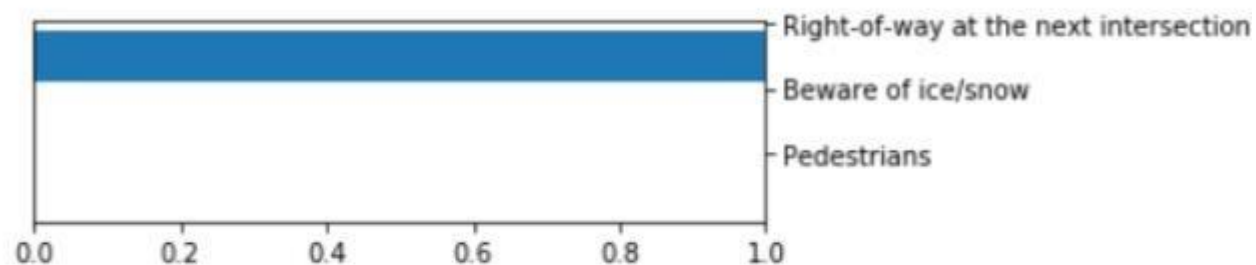
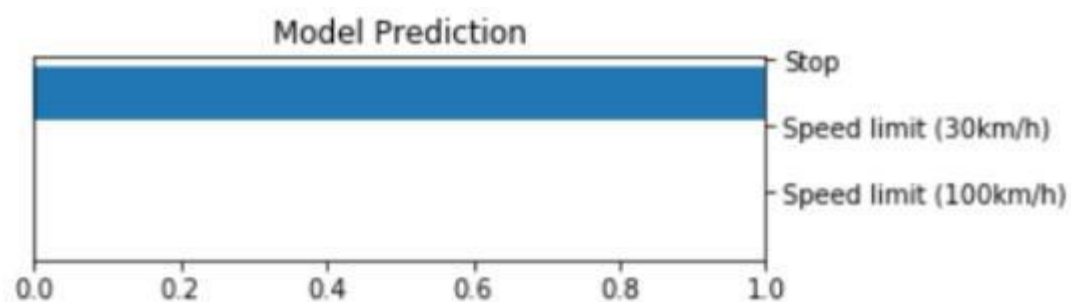
Left Camera Image



Center Camera Image



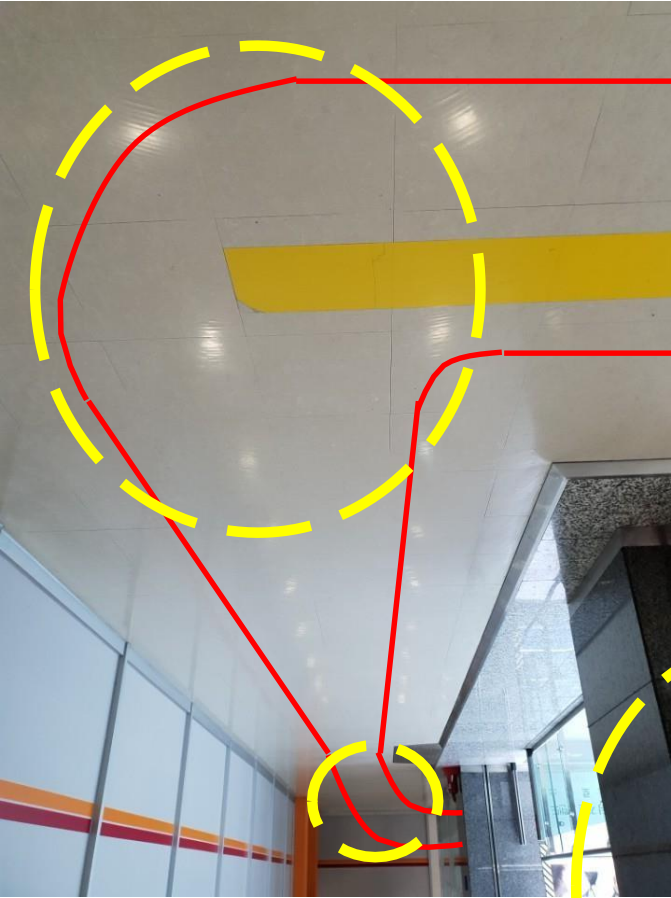
Right Camera Image



***Driving Track***



노란색원(급커브 구간)



신호등



주차

장애물  
(차량)



교통 신호(속도제한)  
PI 제어기 동작 여부 판별

# ***Lidar Scan Strategy***

# Lidar Specification

먼저 사용하려는 Lidar 의 스펙을 살펴보도록 한다.

아래는 사용하는 Lidar 의 데이터시트에서 발췌한 내용에 해당한다.

최대 사정 거리가 40 m 이며 정확도는 5 m 미만 구간에선 2.5 cm 의 오차를 가지고

5 m 이상의 구간에선 10 cm 정도의 오차가 발생하기 시작한다.

업데이트 비율은 결국 Target 으로부터 반사되어 돌아오는 신호를 의미하는데 270 Hz 에 해당한다.

## Performance

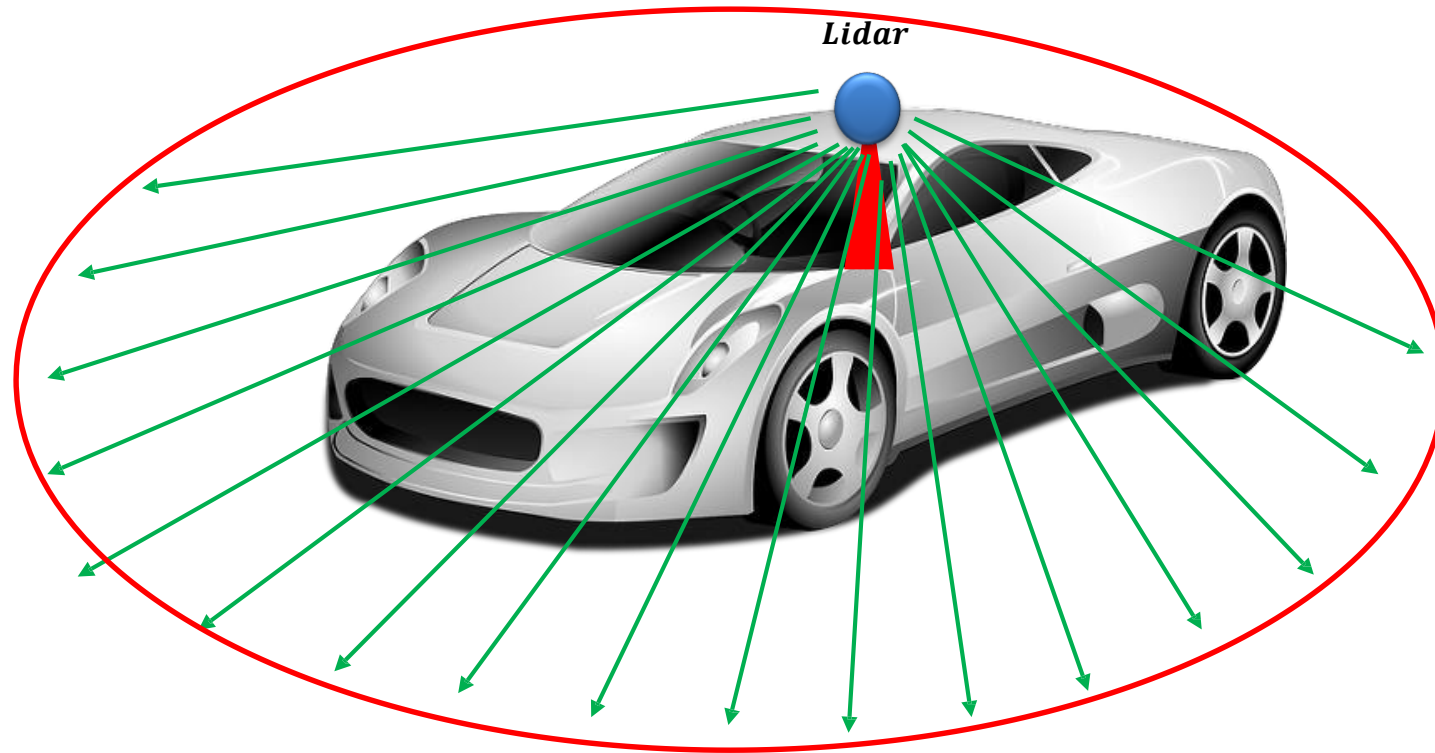
Specification	Measurement
Range (70% reflective target)	40 m (131 ft)
Resolution	+/- 1 cm (0.4 in.)
Accuracy < 5 m	±2.5 cm (1 in.) typical*
Accuracy ≥ 5 m	±10 cm (3.9 in.) typical Mean ±1% of distance maximum Ripple ±1% of distance maximum
Update rate (70% Reflective Target)	270 Hz typical 650 Hz fast mode** >1000 Hz short range only
Repetition rate	~50 Hz default 500 Hz max

\*Nonlinearity present below 1 m (39.4 in.)

\*\*Reduced sensitivity

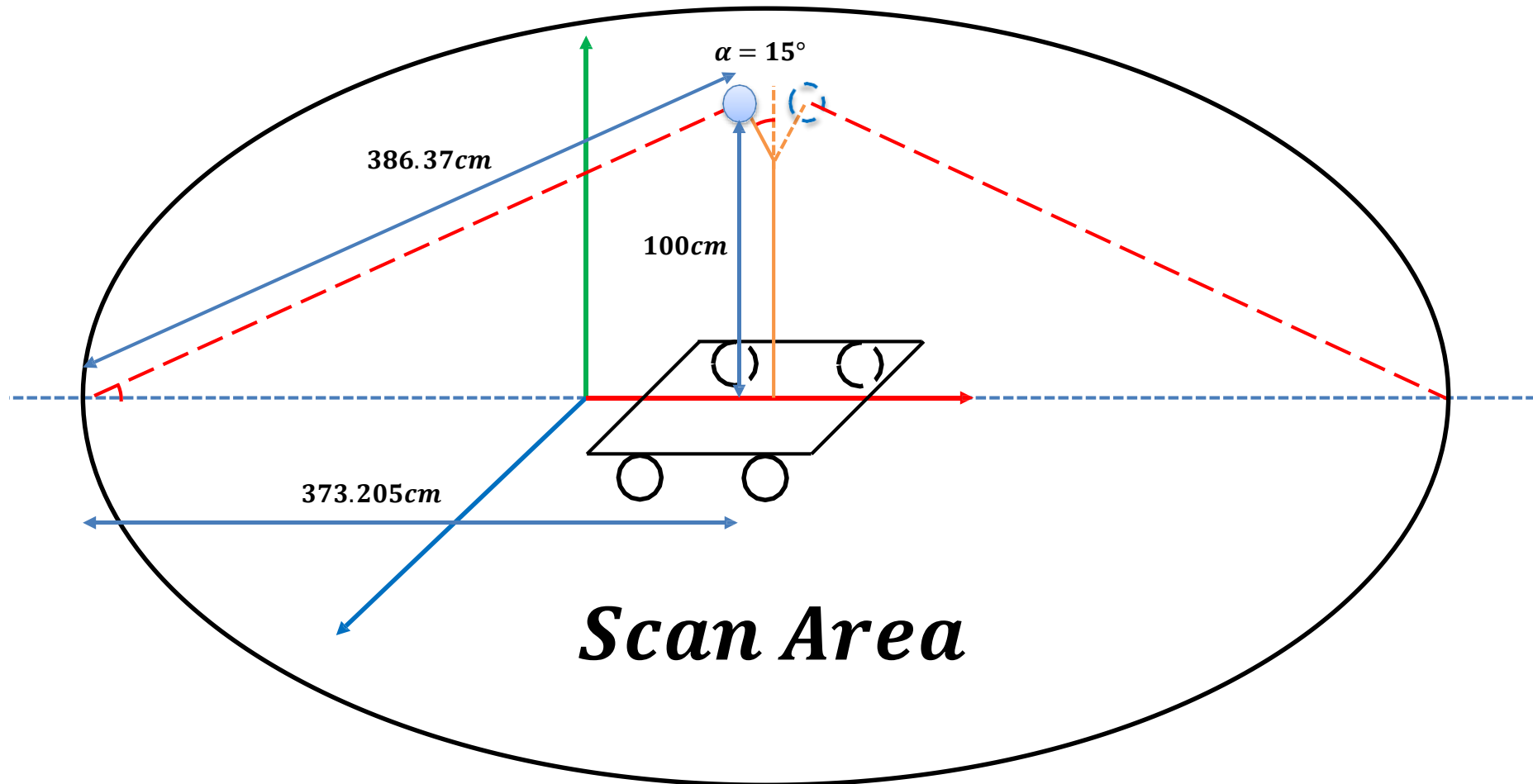
차량과 Lidar 를 배치하여 범위에 대해 모델링 해보도록 한다.

Lidar 를 일정한 각도로 기울여서 그 주변을 계속 스캔하도록 만든다.  
그렇게 되면 일정한 원 구간이 발생하게 되고 그 영역이 Lidar 의 스캔 영역이 된다.  
이 영역 내에 물체가 있는지 없는지를 판별하여 영상과 함께 차량을 제어하는데 활용 된다.



기하학적 해석을 수행하도록 한다.

$$386.37\text{cm} \times \sin(15) \approx 100\text{cm}$$
$$386.37\text{cm} \times \cos(15) \approx 373.205\text{cm}$$



# Lidar Scan Speed Calculation

Lidar 가 얼마나 회전해야 하는지 파악해보도록 한다.

우선 필요한 스펙들을 짚욱 열거해본다.

$$h = 1m$$

$$r = 3.732m$$

$$v_{max} = 3m/s$$

$$Lidar\ Update\ Frequency = 270Hz$$

$$Lidar\ Update\ Period = \frac{1}{270} = 0.0037037037 \dots$$

$$x\ rad/s = \frac{x}{2\pi}\ rev/s = \frac{60}{2\pi} x\ RPM$$

이 상태에서 최소 조건을 걸어보자!

Lidar 의 스캔 각을 15 도라고 가정하고 해석을 진행해보도록 한다.

$$15^\circ : 1 = 360^\circ : x \Rightarrow 360 = 15x$$

$$\therefore x = 24$$

0.0037037037... 초 마다 15 도를 회전하니 0.0037037037 ... 에 24 를 곱하면 1 바퀴 회전하는데 걸리는 시간이다.

$$0.0888888 \dots = \frac{24}{270}$$

주파수 횟수만큼 반복되면 1 초가 되므로 위의 역수는 결국 초당 회전수가 된다.

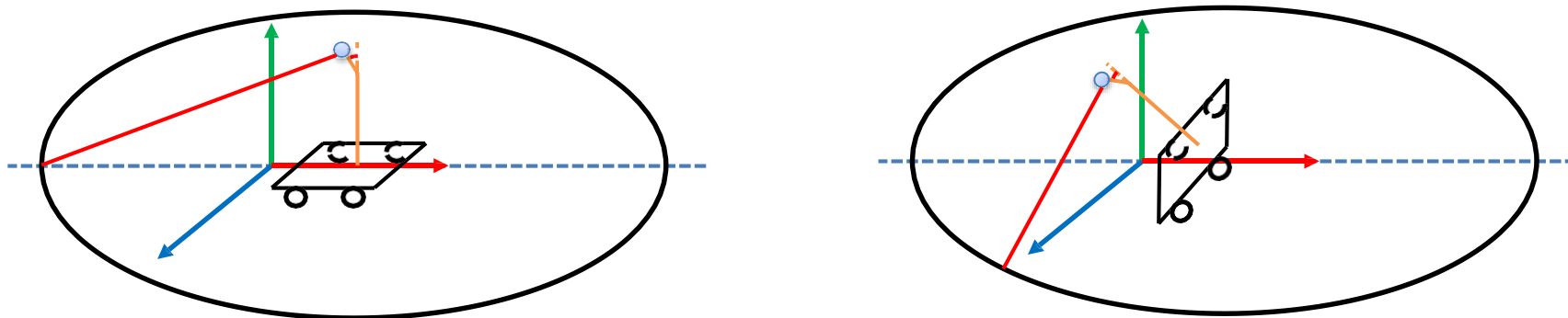
$$\frac{270}{24} = 11.25\ rev/s = 70.68\ rad/s = 675\ RPM$$

# Rotation of the Coordinate System

## 좌표계가 회전한다고?!

우리가 Lidar 를 제어함에 있어서 좌표축 자체의 회전에 대해 신경을 많이 써야 한다.  
예로 좌측의 그림을 보면 현 시점에 Lidar 가 스캔하고 있는 영역은  
1 사분면을 지나 2 사분면과 3 사분면의 경계선인 180 도 영역에 해당한다.

반면 차량 자체가 회전을 하게 될 경우를 고려해보도록 하자!

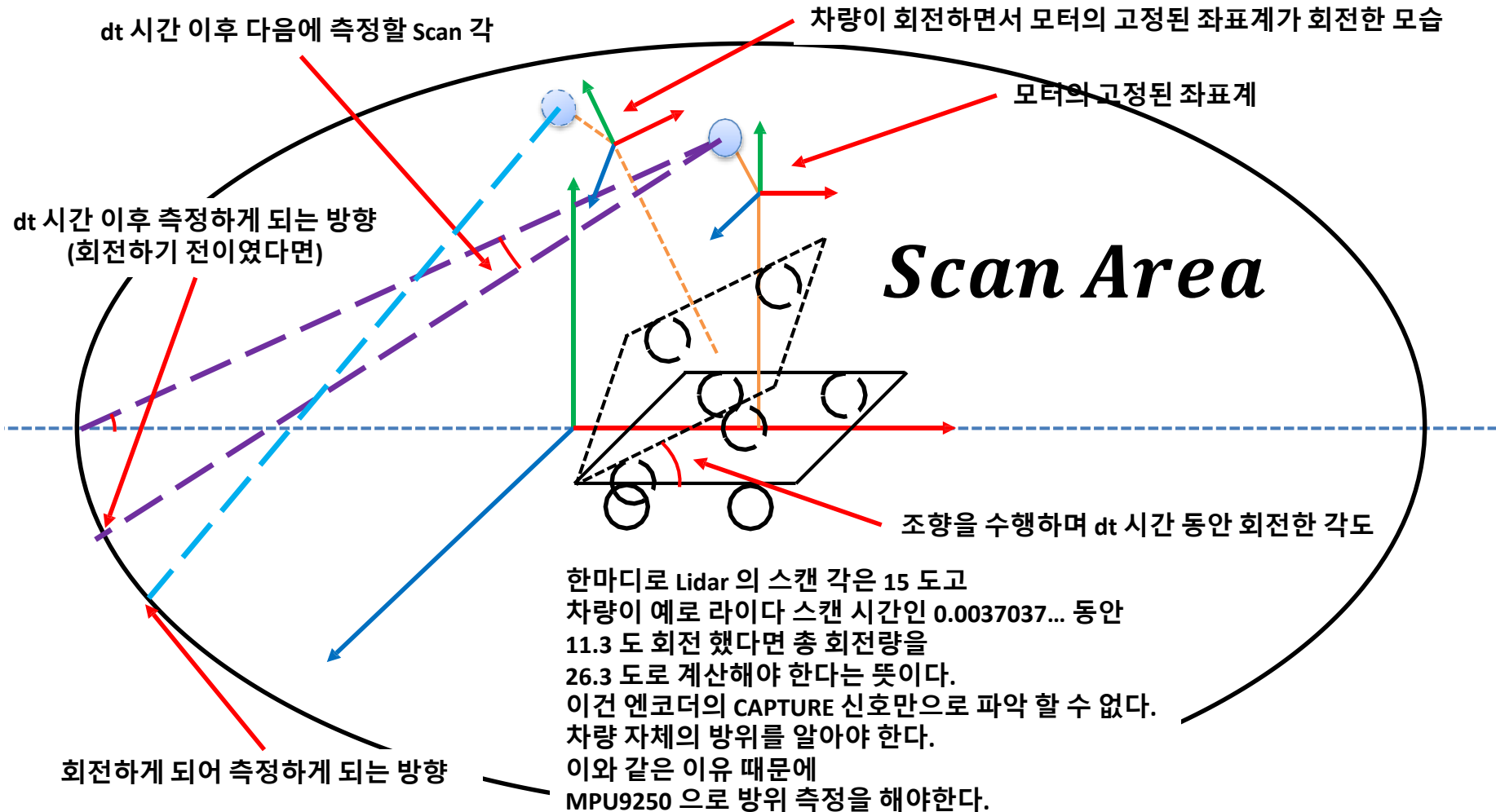


바로 우측 케이스에 해당하는데 차량이 90 도 회전해서 파랑색 축 방향을 향해 달려가고 있다.  
모터는 계속 제자리에서 돌고 있었지만 차량 자체가 90 도 회전을 했기 때문에  
모터의 좌표축 자체가 회전하게 된다는 뜻이다.  
그렇기 때문에 틀어진 만큼 위상을 보정해줘야 한다.  
좌측과 같은 상황이었다는 가정하에 차량이 90 도 틀었다면  
Lidar 는 파랑색 축 방향 즉 270 도 를 스캔하고 있었을 것이다.  
(만약 여기서도 180 도로 계산했다면 잘못된 정보를 가지고 연산을 하고 있으니 분명히 어딘가에 가서 박게 될 것이다)

## Practical Application

## 실 세계에서 적용을 한다면 어떨까?

모터가 675 RPM 으로 돌고 있으니 차량이 회전하는 90 도의 속도보단 당연히 모터가 회전하는 속도가 훨씬 빠르다. 물론 그렇지 않다면 직선 주행도 똑바로 못하고 장애물에 쳐박을 것이다.





# ***Additional Peripheral Features***

## MCU 상에 추가해야할 Peripheral 리스트들

1. 충돌 경보용 사이렌 – 수업 시간 내용 진행(당시 하라고 했었는데 소자 없어서 못한다고 했던것 진행)
2. 우측 좌측 방향 지시등 – 방향 지시등(깜박이 회로) OC 활용
3. LCD 베이스 속도 표시계 – LCD 에 현재 계산한 속도 출력
4. 온도 베이스 선풍기 트리거 – 온도 센서 혹은 MPU6050 에 있는 온도 센서로 온도 떨어지면 구동하게함
5. 전조등 – 있는거 선 빼서 MCU 수업때 진행했던 OC 로 처리
6. 후면등 – 없는거 같아서 전력 전용 LED 활용해서 Peripheral 구성해야할 것임

***Can Custom Protocol***

## CAN 으로 보내야할 것들

조향 신호(좌, 우, 전진, 후진), 가속 신호, 후진 신호, 경보 신호, 깜빡이 신호(좌, 우), 전조등 신호, 영상 데이터 정보, 레이더 데이터 정보, 지정 속도

신호 종류	신호 번호	후속 데이터
좌회전	1	없음
우회전	2	없음
전진	3	없음
후진	4	없음
정지	5	없음
충돌 경보	6	없음
좌측 깜빡이	7	없음
우측 깜빡이	8	없음
전조등 신호	9	없음
영상 데이터	10	장애물 존재 여부(0, 1), 근접 여부
레이더 데이터	11	장애물의 방향(각도), 거리
속도 지정	12	특정 속도값 or PWM Duty
각도 지정	13	특정 각도값 or PWM Duty(서보)

## CAN 으로 보내야할 것들

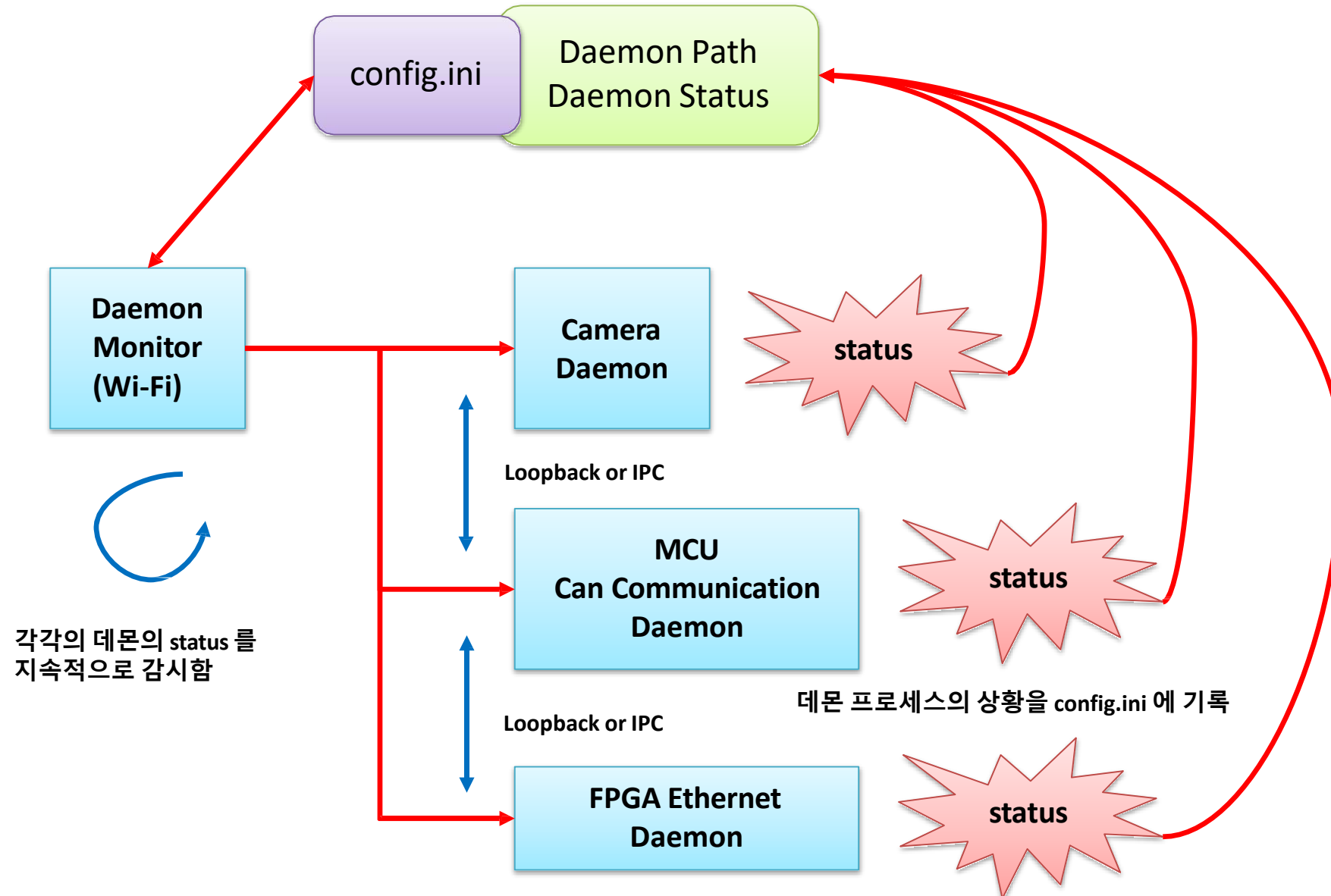
조향 신호(좌, 우, 전진, 후진), 가속 신호, 후진 신호, 경보 신호, 깜빡이 신호(좌, 우), 전조등 신호, 영상 데이터 정보, 레이더 데이터 정보, 지정 속도

신호 종류	신호 번호	후속 데이터
좌회전	1	없음
우회전	2	없음
전진	3	없음
후진	4	없음
정지	5	없음
충돌 경보	6	없음
좌측 깜빡이	7	0 또는 1
우측 깜빡이	8	0 또는 1
전조등 신호	9	0 또는 1
영상 데이터	10	장애물 존재 여부(0, 1), 근접 여부
레이더 데이터	11	장애물의 방향(각도), 거리
속도 지정	12	특정 속도값 or PWM Duty
각도 지정	13	특정 각도값 or PWM Duty(서보)

신호 종류	신호 번호	후속 데이터
좌회전	1	없음
우회전	2	없음
전진	3	없음
후진	4	없음
정지	5	없음
충돌 경보	6	없음
좌측 깜빡이	7	0 또는 1
우측 깜빡이	8	0 또는 1
전조등 신호	9	0 또는 1
영상 데이터	10	장애물 존재 여부(0, 1), 근접 여부
레이더 데이터	11	장애물의 방향(각도), 거리
각도 지정	12	특정 각도값 or PWM Duty(서보)
속도 지정	13	특정 속도값 or PWM Duty
카메라 온/오프	14	0 또는 1
수동/자동 모드	15	0 또는 1

# ***DSP Daemon Architecture***

상태값을 보고 구동되어야 하는 데몬 프로세스가 죽었다면 다시 깨워서 살림





# ***Miscellaneous Issues***

## DSP 상에서 이슈

1. 리눅스 시스템 프로그램을 자동화시키는 UI 구성이 필요함(포토샷 마스터들의 아이콘 제작 실력이 필요함 – 잘 부탁함 ^^)  
2 차 통합 테스트(DSP, MCU, 영상, 무선) 상황에서 번거로운 작업이 수 차례 발생함  
터미널에 접속해서 최초로 서버를 한 번 띄워줘야 하는데 이것을 자동화 시킬 필요성이 있음  
또한 Wi-Fi 가 제대로 작동하지 않을 경우 한 번 내렸다가 다시 올라가게 수정해야함  
그리고 값들이 꼬여서 재부팅이 필요할 경우도 존재하므로 이 녀석도 추가가 필요함
2. 위에서 서버 아키텍처에 표현했듯이 데몬들을 구조적으로 잘 관리해야함
3. 2 차 통합 테스트에서 영상 데이터의 확보가 잘 됨을 확인 했으므로 직선 및 곡선 처리를 수행해야함
4. 통계학에 기반하여 딥러닝을 수행해서 표지판 인식이나 장애물 인식을 시켜야함
5. MCU 및 FPGA 에서 넘어오는 데이터들을 취합하여 학습시키는 데몬이 필요함
6. 기능이 전부 안정화 되면 Audio 기능을 추가하여 Eletric Sound 를 만들거나 Radio 재생을 수행함(핵심 기능은 아님)

## MCU 상에서 이슈

1. 현재 조향이 완벽하지 못함
2. DSP 쪽으로 CAN TX 를 통해서 조향에 대한 실제 PWM 값을 DSP 에게 송신해줘야함  
Speed Control 파트는 Sensored 이므로 그냥 PWM 을 넣어도 되지만  
조향 파트는 Feedback 제어를 직접 하므로 실제 제어하는 PWM 량을 확인해야함
3. 전원부에 스위치 구성이 필요함
4. 경사에 따라 적절하게 속도를 제어하도록 기능을 추가해야함
5. 주변 회로 구성

## FPGA 상에서 이슈

1. 보험으로 Ethernet 통신을 뚫어놓긴 했지만 pmodCAN 에 대한 처리가 필요함(SPI 통신) – 데이터 안정성 측면에서 훨씬 좋고 원래는 이렇게 해야함
2. 보험용으로 만든 Ethernet 제어를 자동화 시킬 필요가 있음(부팅시 FPGA 에서 처리할지 DSP 에서 시리얼 데이터 날리는 프로그램을 구성할지 고민)
3. MPU9250 과 Lidar, 그리고 PWM 및 CAP 이 모두 구성 완료되었으므로 비관성 좌표계에 대한 해석을 본격적으로 수행할 필요가 있음

## 기타 이슈

1. Android 휴대폰 앱을 DSP 와의 통신 프로토콜에 적절하게 구성할 필요가 있으며 버튼 형식이 앱이 필요함  
(어제 여러모로 13 1500 누르기가 상당히 번거롭고 힘들었음 ;; 같이 뛰면서 13 백사리 나서 0123 라든지 13 1700 들어가면 부스터 뿜 ;;)
2. 카메라와 라이다를 안정적으로 고정 시켜야함
3. 모든게 뜨겁다 – 오동작 방지를 위해 **쿨러**가 많이 필요하다!!!

# ***Integrated Test Schedule***

## 1 차 통합 테스트 – Success!

- DC-DC 컨버터와 MCU 그리고 주변 회로들을 구성하여 테스트를 수행함
- 당시 데이터를 13 1, 13 2, 13 3, 13, 5 와 같이 넣으면 PWM 이 1100, 1200, 1300, 1500 으로 들어갔음(수정 필요성이 있었음)

## 2 차 통합 테스트 – Success!

- DC-DC 컨버터와 DSP, MCU, 주변 회로들, 영상, 무선을 구성하여 테스트를 수행함
- 당시 데이터는 13 1000, 1231, 1500, 1540 과 같이 직접 PWM 에 넣을 데이터를 통채로 입력함(타이핑 할게 많아서 백사리 나면 노답 – 수정의 필요성)
- 최초 구동되는 서버의 활성화가 필요한데 할 때마다 직접 Command Line 으로 입력하려고 하는 부분이 번거로웠음(UI 구성 필요)
- 재부팅도 몇 차례 진행했는데 이 역시 직접 해줘야해서 번거로웠음(UI 구성 필요)
- 조향 데이터가 입력이 안되서 사람이 직접 돌려야 했음(MCU 코드 추가 및 수정 필요)
- Wi-Fi 도 무조건 한 번 꺾다가 켜지게 구성을 바꿀 필요가 있음(DSP 상에서 수정 필요)
- 카메라도 고정이 안되어 영상 데이터 확보가 힘들었음(정확한 고정이 필요함)
- MCU 전원부 스위치(하드 스탑)와 PWM 신호선을 꺼버리는 (소프트 스탑)으로 비상 정지를 구현한 필요가 있음

## 3 차 통합 테스트 – Preparing ...

- 2 차 통합 테스트의 미비한 부분들을 모두 보완하여 수행하도록 함

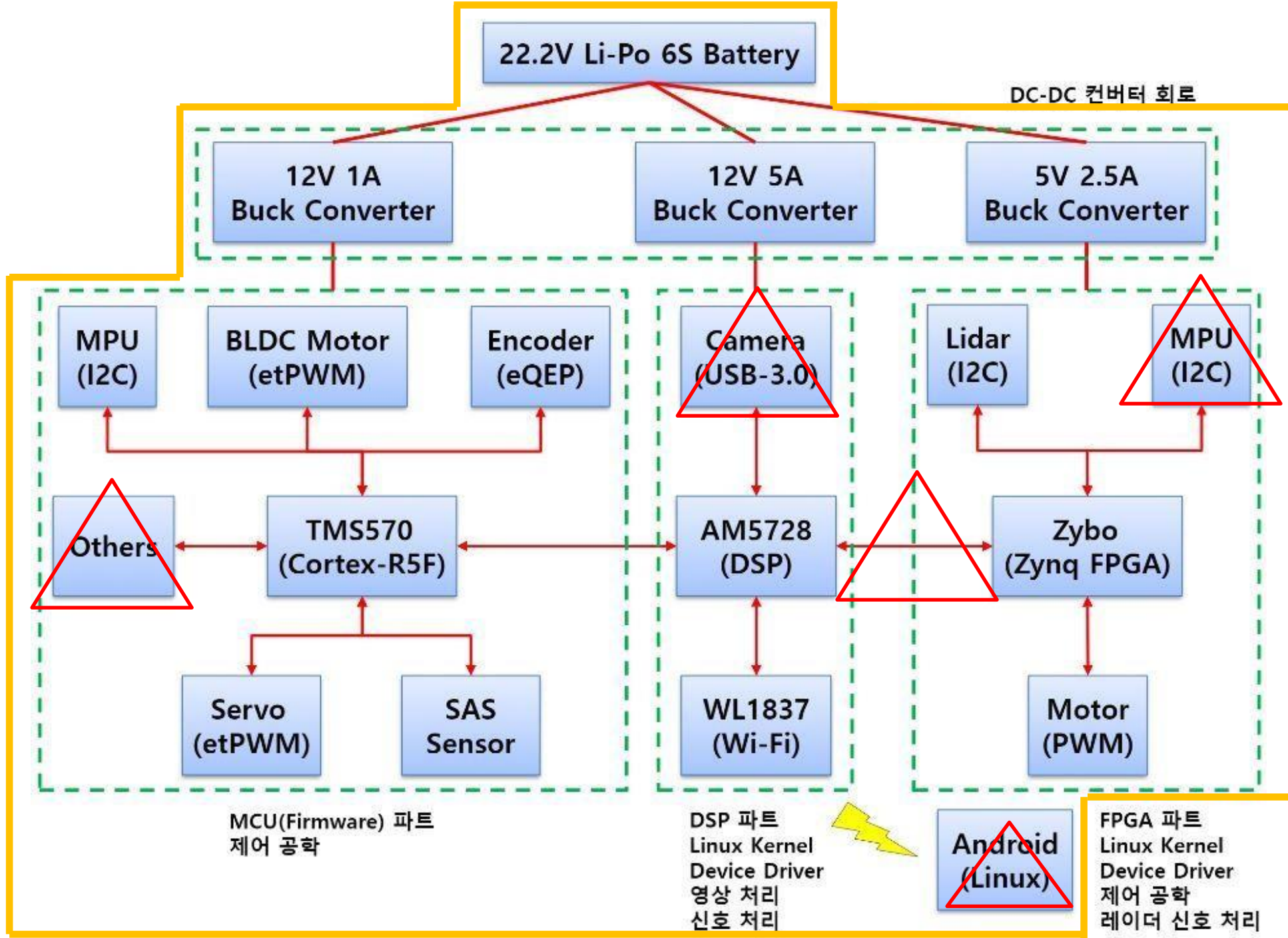
## 4 차 통합 테스트 – Preparing ...

- DC-DC 컨버터와 DSP, FPGA, MCU, 주변 회로들, 영상, 무선과 같은 모든 것들을 구성하여 테스트를 수행함(최종 완성본)

## 5 차 통합 테스트 – Preparing ...

- 인간이 만든 프로그램이 완벽하기는 힘들기에 4 차 통합 테스트에서 미비한 부분들을 수 차례 보완해나가야 할 것임  
(아마도 모든 것들이 다 결합된 상황이기 때문에 보다 다양한 문제점들이 발생할 것으로 생각됨)
- 기존까지 통합 테스트가 RTOS 베이스의 테스트가 아니었으므로 또 어떤 문제가 발생할지 미지수임(항시 고려해야할 부분임)

# ***Current Progress Analysis***



DSP 에서는 추가적으로 Lidar 정보를 OpenGL ES 로 출력해줄 필요가 있다.

FPGA **부팅시 자동**으로 특정 애플리케이션을  
구동시키도록 만들어줘야 한다.

Lidar 를 회전 시키는 작업을 수행하는 모터

Lidar 자체에서 I2C 로 데이터를 가져오는 작업

Lidar 에서 가져온 신호를 가공하여 원하는 결과값을 얻는 작업

Motor 의 회전 속도를 파악하기 위한 Capture 작업

차량의 가속도, 각속도, 방위를 파악하기 위한 MPU9250 작업

DSP 와 통신하기 위한 Ethernet 작업 혹은 SPI CAN 작업

MCU 에서는 나머지 남아 있는 짜잘한 작업들을 All Clear 하도록 한다.

# ***Board Placement***





Power Distribution Board

MCU

LED Cooler

DSP

배터리

FPGA

LED Cooler

