6 조 상태 기반 차량 제어 시스템 결과보고서

6 조 202021025 안준영

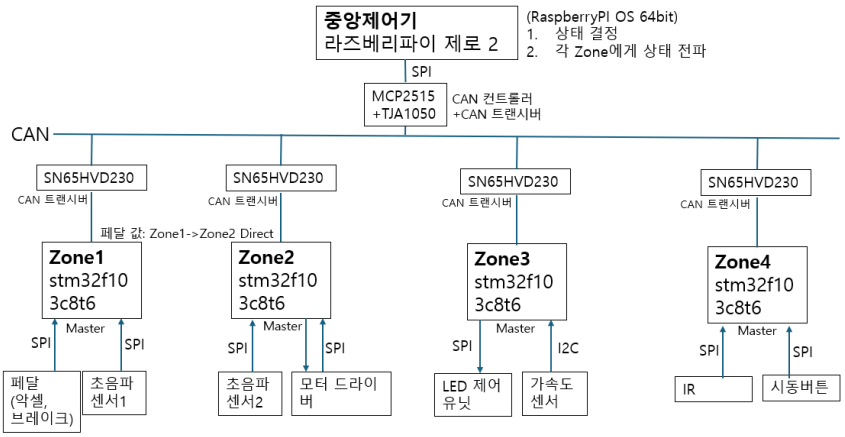
목차

1. 개요

중앙 제어기와 4 개의 Zone 제어기, 총 5 개의 Node 를 갖는 CAN 버스를 통해 상태를 전파하는 시스템이다. 개발 환경은 SSH, STM32CubeIDE 이다. FreeRTOS CMSIS-V2 기반 및 Non-OS 환경에서 STM32F103C8T6 9 개와, RaspberryPi OS 64bit 의 RaspberryPi Zero 2W 를 사용했다. 라즈베리파이는 중앙 제어기로, 시스템 계층구조 하단의 노드로부터 데이터를 제공받아 상태를 결정한다. 총 3 단으로 시스템의 계층을 만들어, 중앙 제어기 - Zone 제어기 - 최하단 제어기로 구성했다. Zone 제어기는 총 4 개로, 각 Zone 산하의 노드들과 중앙 제어기간의 통신을 관장한다.

2. 구성

2.1 구성 소개



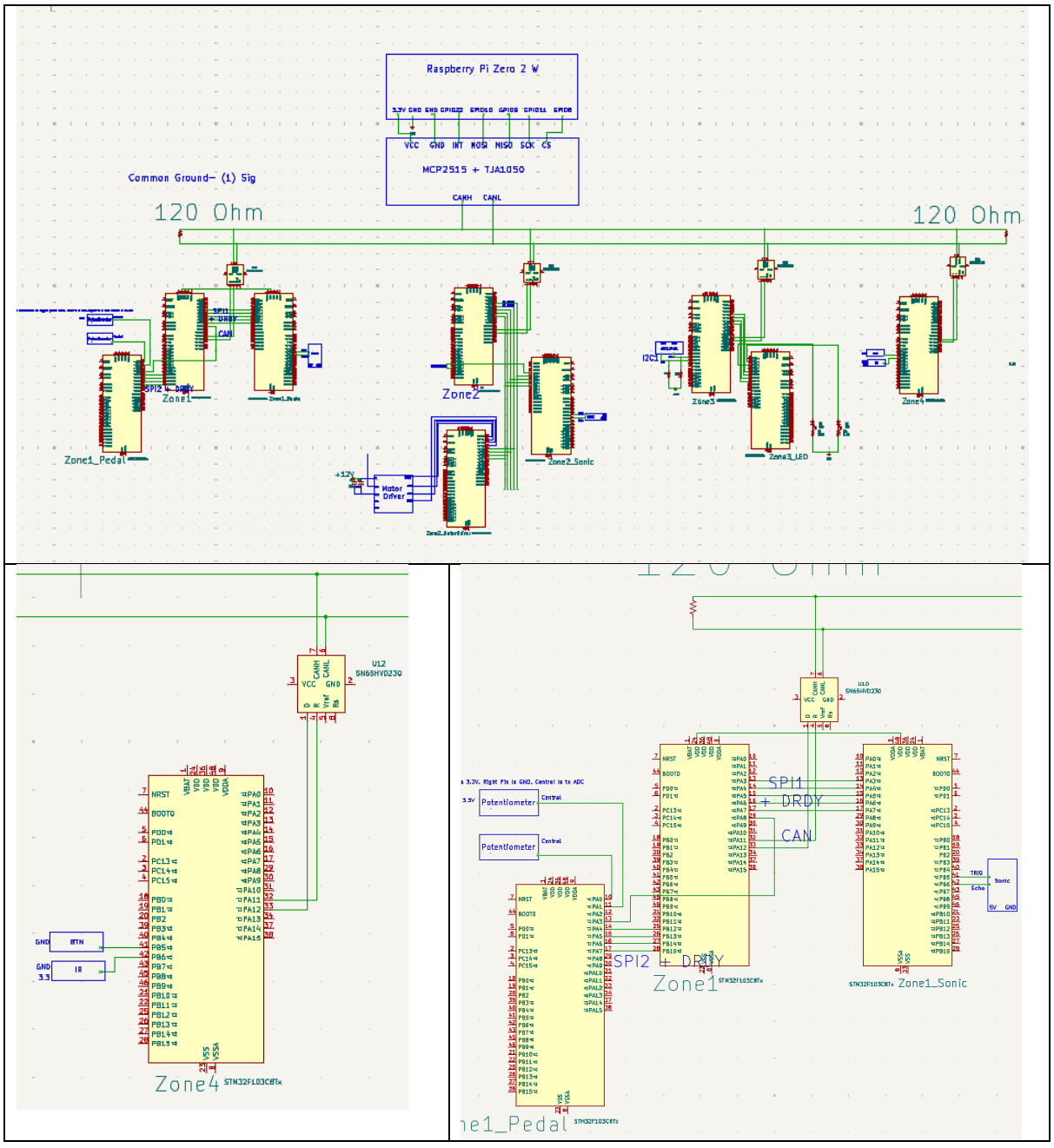
이름	보드 종류	기능
중앙 제어기	Raspberry Pi Zero 2 W	- 상태 결정
Zone1	STM32F103C8T6 호환보드	- 페달 전송 - 초음파 전송
Zone2	STM32F103C8T6 호환보드	- 초음파 전송 - 모터 드라이버 전송
Zone3	STM32F103C8T6 호환보드	- LED 전송 - 가속도 전송
Zone4	STM32F103C8T6 호환보드	- IR 전송 - 버튼 전송

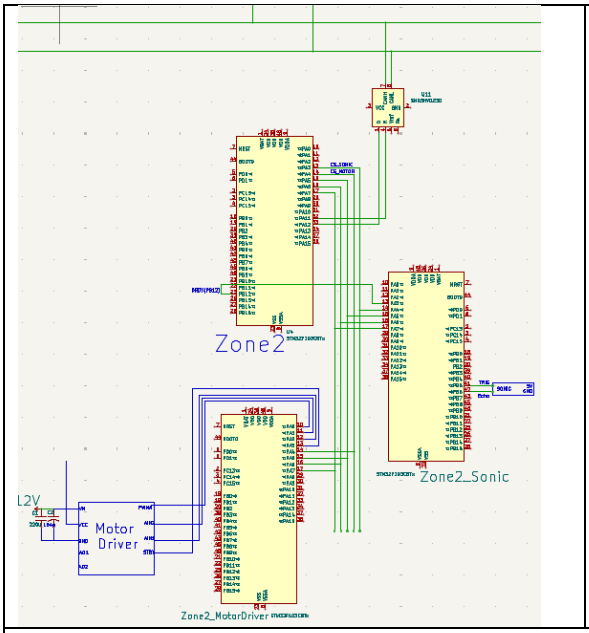
보드 결정 주요 요인은 가격으로, 수업에서 배운 STM32 제품을 쓰되 여러 보드를 써야 하기에 가능한 저렴하지만 적당한 스펙과 TIM 기능이나 통신 인터페이스가 적절히 탑재되어 있는 것을 별도로 선택해 사용했다.

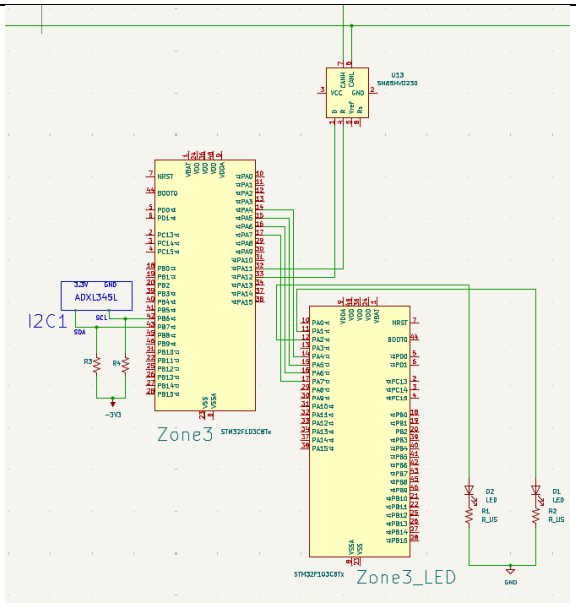
Zone	통신	산하 모듈	기능
Zone1	SPI1	페달 (STM32F103C8T6)	페달로서 2 개의 가변 저항 측정 및 데이터 전송
	SPI2	초음파 (STM32F103C8T6)	HC-SR04 초음파 센서 구동 및 데이터 전송
Zone2	SPI1	초음파 (STM32F103C8T6)	HC-SR04 초음파 센서 구동 및 데이터 전송
	SPI2	모터 드라이버	TB6612FNG 모터 드라이버 구동

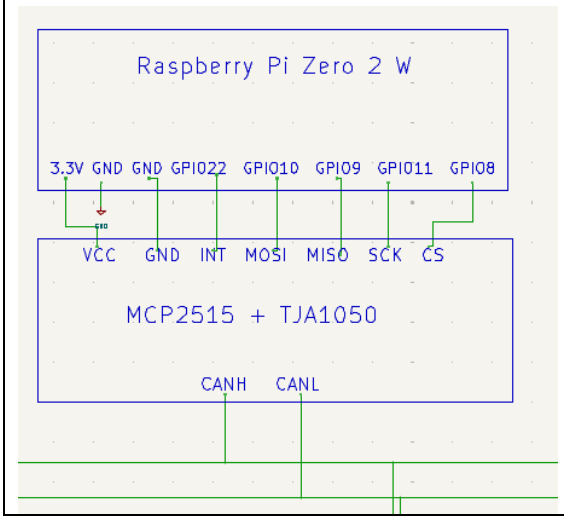
		(STM32F103C8T6)	
Zone3	SPI1	LED 제어 유닛 (STM32F103C8T6)	상태에 따른 LED 제어
	I2C1	가속도	ADXL345 가속도 센서 측정 및 전송
Zone4	SPI1	IR	탑승 감지
	SPI2	BTN	시동 버튼 감지

2.2 배선도



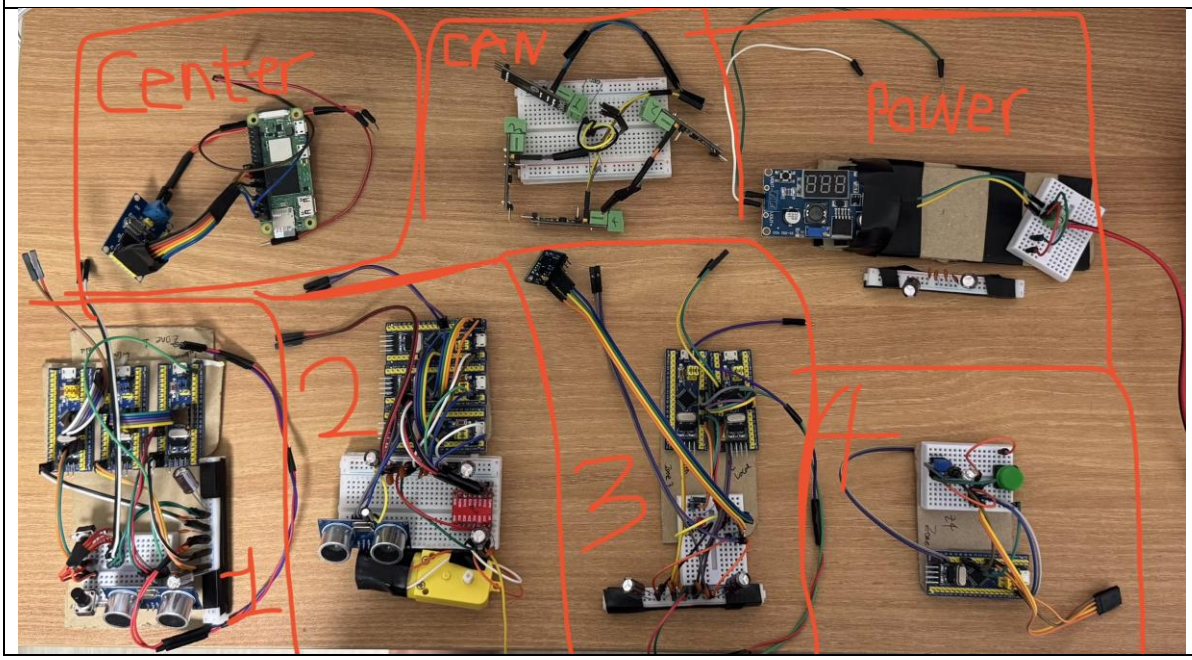

Zone2
Zone2_Sonic
Motor
Driver
Zone2_MotorDriver


I2C1
ADXL345L
Zone3
Zone3_LED


Raspberry Pi Zero 2 W
3.3V GND GND GPIO22 GPIO10 GPIO9 GPIO11 GPIO8
VCC GND INT MOSI MISO SCK CS
MCP2515 + TJA1050
CANH CANL

2.3 실제 구성

전체 구성



3. 기능 설명 및 STATE 설명

3.1 기능 설명 및 STATE 설명

이 시스템은 총 8 개의 상태를 가진다. OFF, READY, DRIVE, EMERGENCY, PEDAL_ERR, MOTOR_ERR, CENTRAL_DOWN 은 중앙 제어기가 결정하는 상태이고, BROKEN_SELF 는 Zone 이 결정하는 상태이다. 중앙 제어기와 Zone 통신 불량 시, 스스로의 통신이 문제라고 Zone 제어기가 판단한 상황을 규정하기 위해 자체적으로 BROKEN_SELF 라고 판정한다. 페달과 모터 고장은 안전과 직결되기에 별도의 에러 상태로 규정했다.

STATE 별 판정 조건을 아래 첨부한다.

상태	조건
OFF	전원 인가후 초기 상태이다.
READY	탑승을 감지한 경우에 OFF 상태에서 READY 상태로 전환한다.
DRIVE	READY 상태에서 시동 버튼이 눌린 경우 DRIVE 로 전환한다.
EMERGENCY	충돌 위험 발생시 DRIVE 에서 전환한다.
PEDAL_ERR	최하단의 페달 노드에 문제가 생겼을 때 발생한다.
MOTOR_ERR	최하단의 모터 드라이버 노드에 문제가 생겼을 때 발생한다.
CENTRAL_DOWN	중앙 제어기의 CAN 이 망가져 통신이 불가할 때 발생한다.
BROKEN_SELF	Zone 의 CAN 이 망가진 경우에 내부에서 발생한다.

STATE 별 동작은 아래와 같다.

1. OFF : IR 센서를 통해 탑승을 감지한다. IR 센서에서 탑승이 감지되었을 때, 중앙 제어기에게 알리고, 중앙 제어기는 READY 로 상태를 전환하고 전파한다.

2. READY : 버튼을 통해 탑승을 감지한다. 버튼 눌림이 감지되었을 때, 중앙 제어기에게 알리고, 중앙 제어기는 DRIVE 로 상태를 전환하고 전파한다.

3. DRIVE : 페달 조작으로 모터를 구동한다. 초음파, 가속도 값으로 충돌 위험을 검사한다.

4. EMERGENCY : 모터는 비상 정지한다. 가속도 값을 측정해, 정지 상태를 판단하고 정지 상태라면 다시 READY 상태로 전환된다. Recovery 가 가능한 상태이다.

5. PEDAL_ERR : 중앙 제어기에서 비상 정지 명령을 내린다. Recovery 가 불가한 상태이다.

6. MOTOR_ERR : 모터 고장 상태를 알린다. Recovery 가 불가한 상태이다.

7. CENTRAL_DOWN : 중앙 제어기가 고장나도, 수동 정지를 하는 등 운행이 가능해야한다. Zone1 에서의 페달 데이터 전송을 제외하곤 모든 CAN 메시지 전송을 중단한다.

8. BROKEN_SELF : Zone 내부에서 발생한 CAN 고장 상태로, CAN 을 중단한다.

3.2 에러 상황 및 대응

발생하는 에러를 에러 주체, 에러 원인, 에러 대응으로 정리해 각각을 설계했다. 아래 표를 통해 확인할 수 있다.

		에러 주체				
		중앙 제어기	**Zone1**	**Zone2**	**Zone3**	**Zone4**
에러 원인	CAN 하트비트/ 송신	CENTRAL_DOWN1	BROKEN_SELF1 EMERGENCY 1	BROKEN_SELF3 EMERGENCY 3	Isolation 3	Isolation 6
	CAN 데이터 변조	CENTRAL_DOWN2	EMERGENCY 2	EMERGENCY 4	Isolation 4	Isolation 7
	SPI 에러		PEDAL_ERR or Isolation1	MOTOR_ERR or Isolation2	Isolation 5	

- MOTOR_ERR : Zone2 의 모터 드라이버 담당 MCU 와의 SPI 통신 불가 상태이다.

(1) Zone1 은 CAN 으로 페달 데이터를 전송하는 기능을 제외한 나머지 기능들을 유지한다.

(2) Zone2 는 CAN 으로 ID_2_MOTOR_SPI_ERR 로 발송하고, 모터 드라이버에게 명령하는 기능을 제외한 나머지 기능들을 유지한다.

(3) Zone3 는 LED 에게 MOTOR_ERR 를 나타내는 명령을 내리고 나머지 기능들을 유지한다.

(4) Zone4 는 기능을 유지한다.

(5) 중앙 제어기는 ID_2_MOTOR_SPI_ERR 를 받고, 이를 로깅한 후 CAN 으로 전파한다.

(6) Zone2 의 모터 드라이버 담당 MCU 는 SPI 불가 시 모터를 정지한다.

- PEDAL_ERR : Zone1 의 페달 담당 MCU 와의 SPI 통신 불가 상태이다.

(1) Zone1 은 CAN 으로 ID_1_PEDAL_SPI_ERR 로 발송하고, 페달 데이터 CAN 전송을 제외한 나머지 기능을 유지한다.

(2) Zone2 는 모터 정지 명령을 내리고 나머지 기능을 유지한다.

(3) Zone3 는 LED 에게 MOTOR_ERR 를 나타내는 명령을 내리고 나머지 기능들을 유지한다.

(4) Zone4 는 기능을 유지한다.

(5) 중앙 제어기는 ID_1_PEDAL_SPI_ERR 를 받고, 이를 로깅한 후 CAN 으로 전파한다.

(6) Zone1 의 페달 MCU 는 계속해서 전송을 시도한다.

- EMERGENCY : 비상 상황이다. 이후 상세 서술한다.

에러외에도, 초음파 거리 위험 및 가속도 센서 값 위험으로도 발동된다.

(1) Zone1 : 하트비트를 제외한 나머지 기능을 정지한다.

(2) Zone2 : 모터 정지 명령을 내리고, 하트비트만을 CAN 으로 보낸다.

(3) Zone3 : 가속도 값, 하트비트는 계속 내보내고 LED 에게 EMERGENCY 를 나타내게 한다.

(4) Zone4 : 버튼 대기 상태로 들어간다.

(5) 중앙 제어기 : 하트비트 및 가속도 센서 값을 감시한다. 가속도 값이 정지 상태를 나타내고, 각 Zone 들로부터 이상 상태가 보고 되지 않으면 다시 READY 상태로 전환시킨다.

- CENTRAL_DOWN

각 존 내부에서 자체적으로 결정하는 상태이다. 중앙 제어기가 죽었다고 판정한다.

(1) Zone1: 페달 데이터 송신을 제외한 나머지 CAN 기능을 정지한다.

(2) Zone2: 페달 데이터 수신 및 모터 드라이버 명령을 제외한 나머지 기능을 정지한다.

(3) Zone3: LED 명령으로 중앙 제어기 다운을 알리는 기능을 제외하고 기능 정지한다.

(4) Zone4: 기능을 정지한다.

- Isolation 구분

(1) Sonic Isolation - Zone1

중앙 제어기 : 충돌 감지 소스에서 초음파 1 을 제외한다.

Zone1 : SPI1 Recovery 를 수행한다.

(2) Sonic Isolation - Zone2

중앙 제어기 : 충돌 감지 소스에서 초음파 2 를 제외한다. 두 개를 배치해놨기에 Fault-Tolerance 를 갖는다.

Zone2 : SPI1 Recovery 를 수행한다.

(3) ACC Isolation - Zone3

중앙 제어기 : 충돌 감지 소스 및 차량 정지 판정에서 가속도 센서를 제외한다. 즉, EMERGENCY 발생 후 READY 전환 기능을 제외한다.

Zone3 : I2C1 을 제외한다.

(4) LED Isolation - Zone3

중앙 제어기 : 아무 동작 하지 않는다.

Zone3 : SPI1 Recovery 를 수행한다.

(5) IR Isolation - Zone4

중앙 제어기 : 아무 동작하지 않는다.

Zone4 : 아무 동작하지 않는다.

(6) BTN Isolation - Zone4

중앙 제어기 : 아무 동작하지 않는다.

Zone4 : 아무 동작하지 않는다.

(7) Zone3 Isolation - Zone3 CAN 고장

중앙 제어기 : 충돌 감지 소스 및 차량 정지 판정에서 가속도 센서를 제외한다. 즉, EMERGENCY 발생 후 READY 전환 기능을 제외한다. Zone3 를 CAN 에서 제외한다.

(8) Zone4 Isolation - Zone4 CAN 고장

중앙 제어기 : Zone4 를 CAN 에서 제외한다.

3.3 에러 감지 및 판정

기본적으로 에러는, 단일 에러 발생 시 에러 판정이 아닌 카운터를 도입해 일정 횟수 이상 연속 발생해야 판정 내리도록 해 노이즈에 내성을 갖도록 했다.

(1) CENTRAL_DOWN1

각 존에서 중앙 제어기의 하트비트를 감시한다. 예외로, OFF 상태 즉 시스템 꺼져있는 상태에선 하트비트를 감시하지 않는다.

(2) CENTRAL_DOWN2

각 존에서 RxFifo0MsgPendingCallback 에서 체크섬, 카운터(시퀀스 필드)를 검증한다. ERR_TYPE_CHKSUM, ERR_TYPE_RX_STUCK 으로 구분 가능하지만 일단 통합해서 처리한다.

(3) BROKEN_SELF1 & EMERGENCY1

Zone1 내부에서, CAN HW 오류 등 송신 불가에 대한 콜백으로 감지한다. 혹은 중앙 제어기에서 ZONE1 의 HEARTBEAT 를 일정시간 이상 감지하지 못할 때 판정한다.

BROKEN_SELF 는 Zone 내부에서 전환되는 상태이고, Zone1 다운은 EMERGENCY 발생 조건이다. EMERGENCY 에서는 PEDAL 데이터를 무시하고 강제로 0 으로 모터에 반영하므로 안전하다.

(4) EMERGENCY2

Zone1 이 보내는 패킷에 오염이나 시퀀스 불일치가 연속될 때 이를 중앙제어기에서 감지후 발생한다. 중앙 제어기에서 이 상황에서 Zone1 다운으로 취급하고 EMERGENCY 를 전파한다.

(5) BROKEN_SELF2 & EMERGENCY3

Zone2 내부에서, CAN HW 오류 등 송신 불가에 대한 콜백으로 감지한다. 혹은 중앙 제어기에서 ZONE2 의 HEARTBEAT 를 일정시간 이상 감지하지 못할 때 판정한다. BROKEN_SELF 에서는 Zone2 내부에서 자체적으로 모터를 정지시킨다. 중앙 제어기에서는 EMERGENCY 를 전파한다.

(6) EMERGENCY4

Zone2 가 보내는 패킷에 오염이나 시퀀스 불일치가 연속될 때 이를 중앙제어기에서 감지후 발생한다. 중앙 제어기에서 이 상황에서 Zone2 다운으로 취급하고 EMERGENCY 를 전파한다. Zone2 는 반사 ACK 로 이 상황을 감지하기 보단, 버스 점유율을 고려해 단순히 중앙제어기의 하트비트에 카운터(시퀀스)를 담고 이를 검증해 감지한다. 따라서, CENTRAL_DOWN 체킹 로직에 걸려 CENTRAL_DOWN 으로 들어간다. 이때, 다른 Zone 들은 EMERGENCY 에 빠져있기 때문에, Zone1 이 계속해서 페달 값으로 0 을 보내 모터가 정지하게 된다.

(7) PEDAL_ERR zone1

Zone1 의 페달 MCU 와 통신하는 SPI2 가 망가진 경우 판정된다. Zone1 의 spi1Task 에서 패킷을 검사해 연속적인 에러발생시 판정한다. 혹은, DRIVE 상태인데 DRDY 신호가 1 초간 오지 않으면 바로 사망판정 한다. 혹은 받은 버퍼의 STATUS 를 통해 알 수 있다. Zone2 는 CAN 에 ID_2_MOTOR_SPI_ERR_ID 로 중앙에 발송해 상태를 알리고, 중앙 제어기는 PEDAL_ERR 를 발령한다.

(8) MOTOR_ERR

Zone2 의 모터 드라이버 MCU 와 통신하는 SPI2 가 망가진 경우 판정된다. 패킷 검사를 통해 알 수 있다. 왜냐면, 버퍼가 이상하게 채워지거나 변하지 않으면 상대방이 다운되서 시퀀스 등이 바뀌지 않기 때문이다. -> Multi duplex 쓰는 이유

(9) ISOLATION 1

Zone1 의 초음파 MCU 와 통신하는 SPI1 이 망가진 경우 판정된다. Zone1 의 내부에서 내부에서 큐를 통해 태스크가 구동하기 때문에, 초음파 데이터가 오지 않는다면 자동적으로 초음파 데이터 CAN 송신은 이루어지지 않는다. 중앙 제어기에서는 DRIVE 상태에서 일정 시간 이상 초음파 데이터가 오지 않으면 데이터를 충돌 로직에서 제외한다.

(10) ISOLATION 2

Zone2 의 초음파 MCU 와 통신하는 SPI1 이 망가진 경우 판정된다. Zone2 의 내부에서 초음파 데이터 CAN 송신을 중단한다. 중앙 제어기에서는 DRIVE 상태에서 일정 시간 이상 초음파 데이터가 오지 않으면 데이터를 충돌 로직에서 제외한다.

(11) ISOLATION 3

중앙 제어기에서 Zone3 의 하트비트를 감시해, 일정 시간 이상 오지 않을 때 발동된다.

ZONE3 뜯어버려

(12) ISOLATION 4

중앙 제어기에서 Zone3 의 패킷을 검사해 검증해낸다. Zone3 측에서는 중앙 제어기의 하트비트 패킷 속 Zone3 CAN 시퀀스 카운터로 검증한다.

(13) ISOLATION 5

Zone3 내부에서 SPI1, I2C 고장시 내부에서 자체적으로 판단한다. CAN 에서 분리시킨다. 가속도 송신시 데이터에 0xFFFF 를 담아 중앙 제어기에게 에러를 알린다. LED 는 플래그만 세우고 별도로 처리하지 않았다.

(14) ISOLATION 6

중앙 제어기에서 Zone4 의 하트비트를 감시해, 일정 시간 이상 오지 않을 때 발동된다.

(15) ISOLATION 7

중앙 제어기에서 Zone4 의 패킷을 검사해 검증해낸다. Zone4 측에서는 중앙 제어기의 하트비트 패킷 속 Zone4 CAN 시퀀스 카운터로 검증한다.