Low-Level Skid Steer Controller

상태: `초안 (Draft)`
버전: `1.0`
작성자: [사용자 이름]
관련 Jira 이슈: `SSV-2`

1. 개요

이 문서는 4WD 스키드 스티어 차량(SSV)의 하위 제어기(TC375)에서 동작할 **로우 레벨 컨트롤러(Low-Level Controller)**의 상세 설계를 정의한다.

이 컨트롤러의 핵심 목적은 상위 제어기(RPi)로부터 수신한 추상적인 차량 목표 움직임 (`v_target`, `w_target`)을 실제 하드웨어인 모터 쉴드를 구동하기 위한 구체적인 제어 신호(`PWM`, `Direction`, `Brake`)로 변환하는 것이다. 또한, 엔코더 센서로부터 물리적인 현재 속도를 추정하여 피드백 제어에 사용한다.

본 설계는 MATLAB/Simulink를 이용한 모델 기반 설계(MBD) 방식으로 구현되며, 최종적으로 TC375에 배포 가능한 C 코드를 생성하는 것을 목표로 한다.

2. 전체 시스템 아키텍처

컨트롤러는 3개의 주요 서브시스템으로 구성된 계층적 구조를 가진다. 각 서브시스템은 명확하게 정의된 역할을 수행하여 전체 시스템의 모듈성과 재사용성을 높인다.

2.1 최상위 인터페이스

- **입력 포트 (Inputs):**
  - `v_target` (double, m/s): 차량의 목표 선속도
  - `w_target` (double, rad/s): 차량의 목표 각속도
  - `enc_pulse_L` (int32, count): 좌측 후륜 엔코더의 누적 펄스
  - `enc_pulse_R` (int32, count): 우측 후륜 엔κο더의 누적 펄스
- **출력 포트 (Outputs):**
  - `pwm_L/R` (uint8): 좌/우측 모터 쉴드의 PWM 신호 (0~255)

- `dir_L/R` (boolean): 좌/우측 모터 쉴드의 방향 신호 (1: 정방향, 0: 역방향)
- `brake_L/R` (boolean): 좌/우측 모터 쉴드의 브레이크 신호 (1: Brake On)

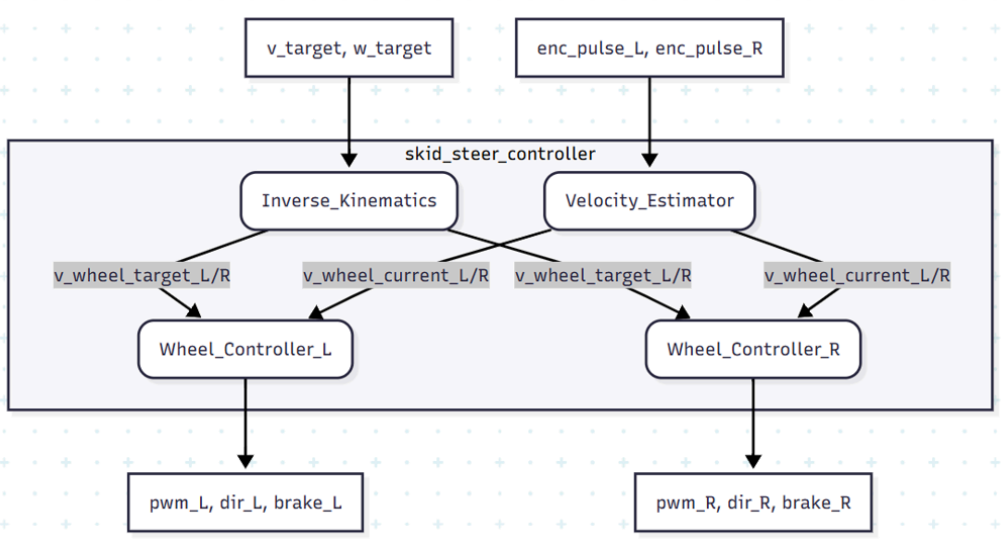
2.2 내부 블록 다이어그램

```
graph TD
    subgraph "skid_steer_controller"
        C(Inverse_Kinematics)
        D(Velocity_Estimator)
        E(Wheel_Controller_L)
        F(Wheel_Controller_R)
    end

    A[v_target, w_target] --> C
    B[enc_pulse_L, enc_pulse_R] --> D

    C -- "v_wheel_target_L/R" --> E & F
    D -- "v_wheel_current_L/R" --> E & F

    E --> G[pwm_L, dir_L, brake_L]
    F --> H[pwm_R, dir_R, brake_R]
```



3. 컴포넌트별 상세 설계

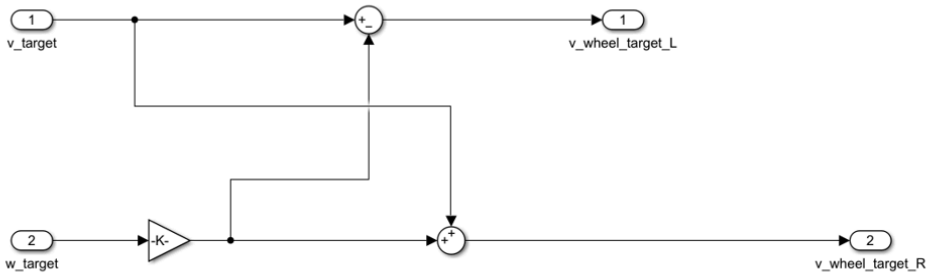
3.1 `Inverse_Kinematics` (역기구학)

- **역할:** 차량 전체의 목표 움직임을 각 바퀴의 개별 목표 속도로 변환하는 '번역가' 역할을 수행한다.
- **수식:**
  - `v_wheel_target_L = v_target - (w_target * TRACK_WIDTH / 2)`

- `v_wheel_target_R = v_target + (w_target * TRACK_WIDTH / 2)`

- **마스크 파라미터:**

이름	설명	단위	기본값
`TRACK_WIDTH`	차량 좌우 바퀴 간의 거리	m	`0.25`

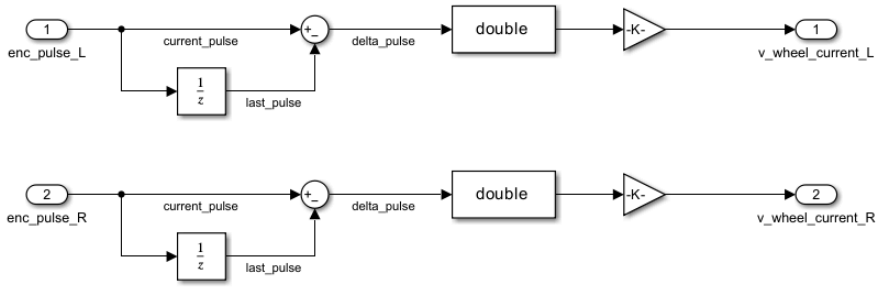


3.2 `Velocity_Estimator` (속도 추정기)

- **역할:** 엔코더로부터 들어오는 누적 펄스 카운트를 m/s 단위의 물리적 속도로 변환하는 '해석가' 역할을 수행한다.
- **수식:**
  - `delta_pulse = current_pulse - previous_pulse`
  - `speed = delta_pulse * (WHEEL_DIAMETER * pi) / (PPR * SAMPLING_TIME)`
- **마스크 파라미터:**

이름	설명	단위	기본값
`PPR`	엔코더가 1회전 할 때 발생하는 펄스 수	count	`400`
`WHEEL_DIAMETER`	바퀴의 지름	m	`0.065`

`SAMPLING_TIME`	컨트롤러의 실행 주기	s	`0.01`



3.3 `Wheel_Controller` (바퀴 제어기)

- **역할:** '목표 바퀴 속도'와 '현재 바퀴 속도'의 오차를 계산하고, 이를 줄이기 위해 모터 쉴드에 보낼 최종 제어 신호를 생성하는 컨트롤러의 '두뇌' 역할을 수행한다.
- **제어 로직:** PI(비례-적분) 제어기를 사용하여 오차를 보상한다. `Discrete PID Controller` 블록을 사용하며, 출력은 `-1.0` (최대 역방향) ~ `1.0` (최대 정방향) 범위로 정규화된다.
- **내부 컴포넌트:** `Arduino_Shield_Driver_Logic`
  - **역할:** PID 컨트롤러의 정규화된 출력(`pid_output`)을 Arduino Motor Shield (L298P)가 이해할 수 있는 3개의 구체적인 신호(`pwm` , `direction` , `brake`)로 변환한다.
  - **동작 테이블:**

`pid_output` (입력)	`pwm_duty` (출력)	`direction` (출력)	`brake` (출력)	설명
`1.0`	`255`	`1` (true)	`0` (false)	최대 정회전
`0.5`	`127`	`1` (true)	`0` (false)	절반 속도 정회전
`0.05`	`12`	`1` (true)	`0` (false)	브레이크 임계값 직전

`0.0` (정지)	`0`	`1` (true)	`1` (true)	브레이크 활성화
`-0.5`	`127`	`0` (false)	`0` (false)	절반 속도 역회전
`-1.0`	`255`	`0` (false)	`0` (false)	최대 역회전