

# ROBOTC Programming

## using MINDSTORMS EV3

*Sangwook Lee*  
*Gyeonggido Office of Education*



저작자를 밝히면 이용이 가능하지만, 영리 목적으로 이용할 수 없으며, 내용을 수정해서도 안 됩니다

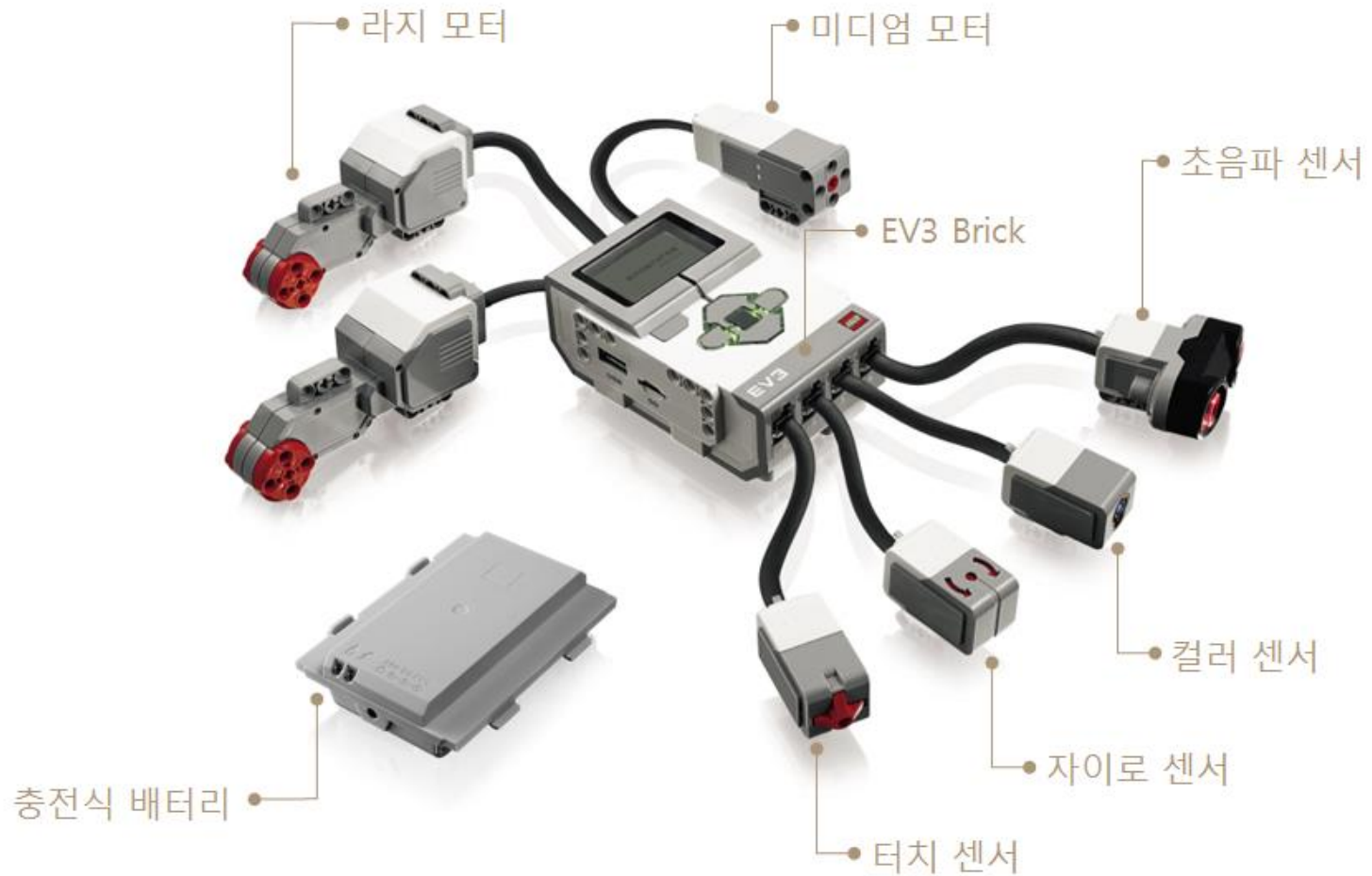
# 04 액추에이터 제어

1. 모터 제어하기

2. 디스플레이, 스피커 사용하기

# 모터와 센서 설정

## • EV3 모터와 센서



# 모터와 센서 설정

- 모터와 센서 설정

- [Robot]-[Motor and Sensors Setup] 선택
- 모터와 센서의 이름, 종류, PID 제어 등을 설정
- 모터와 센서의 이름을 설정하지 않으면 프로그램 작성 시 기본 값을 사용해야 됨
  - 예) motorA, motorB, ...

모터와 센서 설정은...

소스 코드 편집 상태에서 가능

# 모터와 센서 설정

- 표준 모델 EV3\_REMBOT의 모터와 센서 값
  - 모터 값

〈현재 EV3\_REMBOT의 기본 값〉

motorB 이름은 'leftMotor'

motorC 이름은 'rightMotor'

| Standard Models Motors Sensors |            |                   |                          |                                     |                                     |                  |
|--------------------------------|------------|-------------------|--------------------------|-------------------------------------|-------------------------------------|------------------|
| Port                           | Name       | Type              | Reversed                 | Encoder                             | PID Control                         | Drive Motor Side |
| motorA                         | armMotor   | EV3 Motor (Large) | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | None             |
| motorB                         | rightMotor | EV3 Motor (Large) | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Right            |
| motorC                         | leftMotor  | EV3 Motor (Large) | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Left             |
| motorD                         |            | EV3 Motor (Large) | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | None             |

## – 센서 값

| Standard Models Motors Sensors |             |                  |             |
|--------------------------------|-------------|------------------|-------------|
| Sensor Index                   | Name        | Sensor Type      | Sensor Mode |
| S1                             | touchSensor | Touch (EV3)      | Touch       |
| S2                             | gyroSensor  | Gyro (EV3)       | Angle       |
| S3                             | colorSensor | Color (EV3)      | Reflected   |
| S4                             | sonarSensor | Ultrasonic (EV3) | Distance CM |

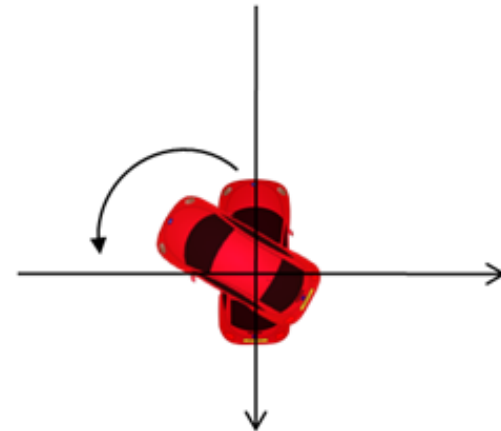
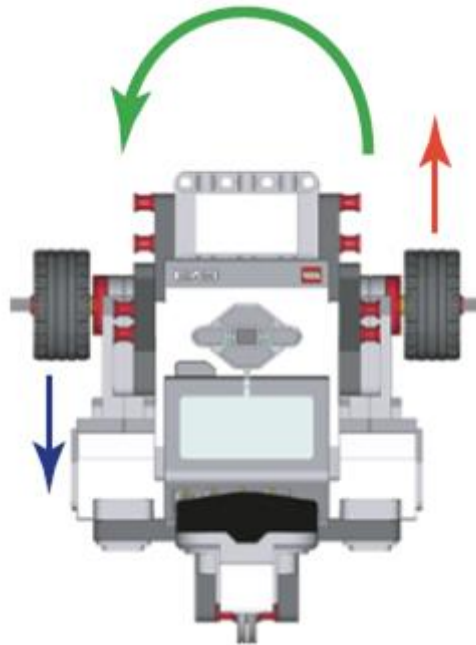
# 모터와 센서 설정

- PID(Proportional Integral Differential) 제어
  - 비례미적분 제어
  - 설정 방법
    - [Motors] 탭에서 PID Control 항목 체크

두 모터를 같은 파워로 가동하더라도  
한쪽 방향으로 약간 쏠려서 진행되는 경우가 있는데  
이런 경우 PID 제어를 설정하면  
좀 더 직선에 가까운 진행이 가능

## [탐구] 포인트 턴(point turn)

- 두 바퀴의 중심을 축으로 하여 3초간 왼쪽으로 회전하는 프로그램을 작성하시오.
  - 파일명: pointturn.c



# [탐구] 포인트 턴(point turn)

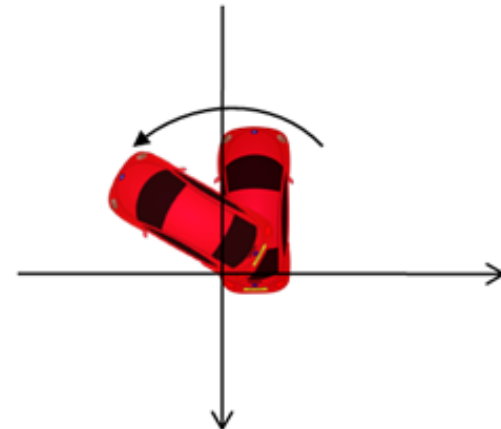
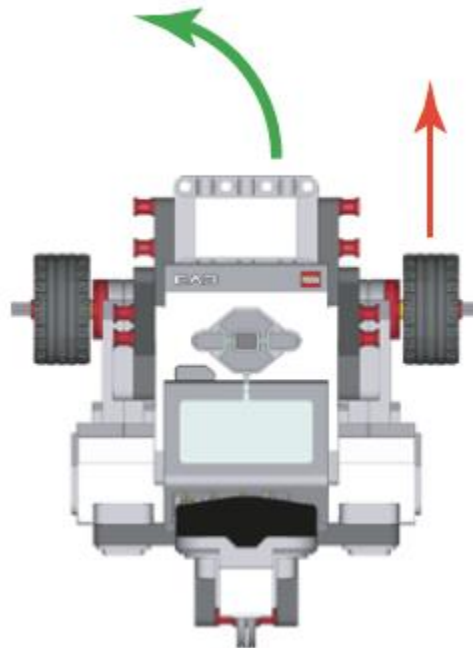
- 코드

```
#pragma config(StandardModel, "EV3_REMBOT")

task main()
{
    setMotorSpeed(leftMotor, -100);
    setMotorSpeed(rightMotor, 100);
    sleep(3000);
}
```

## [탐구] 스윙 턴(swing turn)

- 왼쪽 바퀴를 중심으로 2.5초간 회전하는 프로그램을 작성 하시오.
  - 파일명: swingturn.c



# [탐구] 스윙 턴(swing turn)

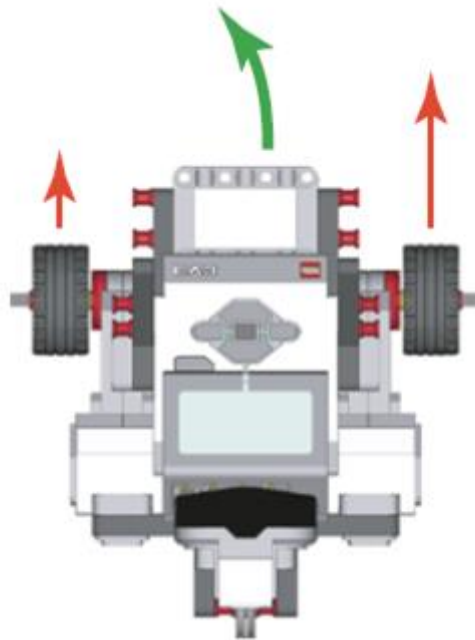
- 코드

```
#pragma config(StandardModel, "EV3_REMBOT")

task main()
{
    setMotorSpeed(leftMotor, 0);
    setMotorSpeed(rightMotor, 100);
    sleep(2500);
}
```

## [탐구] 커브 턴(curve turn)

- 두 바퀴가 같은 방향으로 움직이면서 2초간 왼쪽으로 회전하는 프로그램을 작성하시오.
  - 파일명: curveturn.c



# [탐구] 커브 턴(curve turn)

- 코드

```
#pragma config(StandardModel, "EV3_REMBOT")

task main()
{
    setMotorSpeed(leftMotor, 30);
    setMotorSpeed(rightMotor, 60);
    sleep(2000);
}
```

# 04 액추에이터 제어

1. 모터 제어하기

2. 디스플레이, 스피커 사용하기

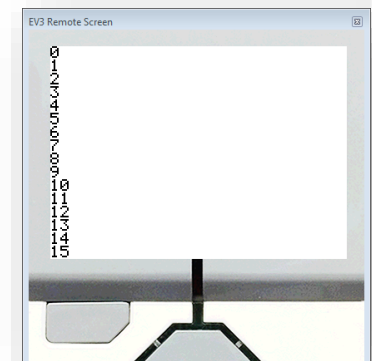
# 디스플레이

- 디스플레이 용도

- 문자열 출력
- 변수 값, 센서 값 출력
  - ☞ 일반적으로 문자열 속에 포함되어 있음
- 점, 선, 원, 사각형 출력

- 디스플레이 해상도

- 가로 178 픽셀, 세로 128 픽셀
- 큰 글자는 가로 16 픽셀, 세로 16 픽셀
- 작은 글자는 가로 8 픽셀, 세로 8 픽셀



# [예제] 초음파 센서 값을 숫자로 출력

- 소스 코드

```
#pragma config(StandardModel, "EV3_REMBOT")

task main()
{
    while(1)
    {
        displayBigTextLine(2, "DISTANCE = %dcm", getUSDistance(S4));
        // LCD 세 번째 줄에, 초음파 센서 값을 나타냄

        sleep(500); // 0.5초마다 바꾸어가며 나타냄
    }
}
```

Q) 전처리 구문 넣을 필요 있나?

# [예제] 초음파 센서 값을 그래프 출력

## • 소스 코드

```
task main()
{
    int dist, pos_x=1;

    while(1)
    {
        if(pos_x>178) → 그래프가 LCD 오른쪽 끝에 도달하면 지우고 처음부터 다시 그리는 선택문
        {
            eraseDisplay();
            pos_x=1;
        }

        dist = getUSDistance(S4); → 센서 값을 dist 변수에 저장
        drawLine(pos_x, 0, pos_x, dist); → 센서 값을 그래프의 높이로 하는 선을 그림

        sleep(100); → 0.1초 간격으로 그림을 그림

        pos_x=pos_x+2;

        }
    }
```

시작 좌표      끝 좌표

# 스피커

- EV3 브릭의 스피커 용도
  - 톤 재생 (주파수로 단일 음 출력)
  - 사운드 파일 재생

## ※ 음의 주파수(Hz)

| 음 | 5옥타브 | 6옥타브 |
|---|------|------|
| 도 | 523  | 1047 |
| 레 | 587  | 1175 |
| 미 | 659  | 1319 |
| 파 | 699  | 1397 |
| 솔 | 784  | 1568 |
| 라 | 880  | 1760 |
| 시 | 988  | 1976 |

# [예제] 동요 '나비야' 연주

## • 소스 코드

```
task main()
{
    int frequency[7]={523, 587, 659, 699, 784, 880, 988}; → 7개의 음을 저장한 배열 선언
    int song[27]={4, 2, 2, 3, 1, 1, 0, 1, 2, 3, 4, 4, 4, 4, 2, 2, 2, 3,
                  1, 1, 0, 2, 4, 4, 2, 2, 2}; → 연주할 음을 순서대로 저장한 배열 선언
    int i, t;
    for(i=0; i<27; i++) → 첫 번째 음부터 마지막 스물일곱 번째 음까지 반복하여 소리 내기 위한 반복분
    {
        t=song[i]; → 연주할 음의 저장 위치를 t에 저장
        playImmediateTone(frequency[t], 50); → 음을 500ms 동안 연주
        if(i==2 || i==5 || i==12 || i==19) → 2, 5, 12, 19번 계이름만 1초 유지
            sleep(1000);
        else
            sleep(500);
    }
}
```

음의 주파수      음의 지속 시간, 단위는 10ms

