

ROBOTC Programming

using MINDSTORMS EV3

Sangwook Lee
Gyeonggido Office of Education



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

05 센서 활용

1. 터치, 초음파, 컬러 센서 활용하기
2. 엔코더 제어하고 활용하기

터치 센서 활용



터치 센서(touch sensor)는...

- 로봇이 사물에 부딪혔을 때를 감지하는데 사용
- 손으로 눌러 원하는 동작을 지시하는 조작 버튼으로 사용

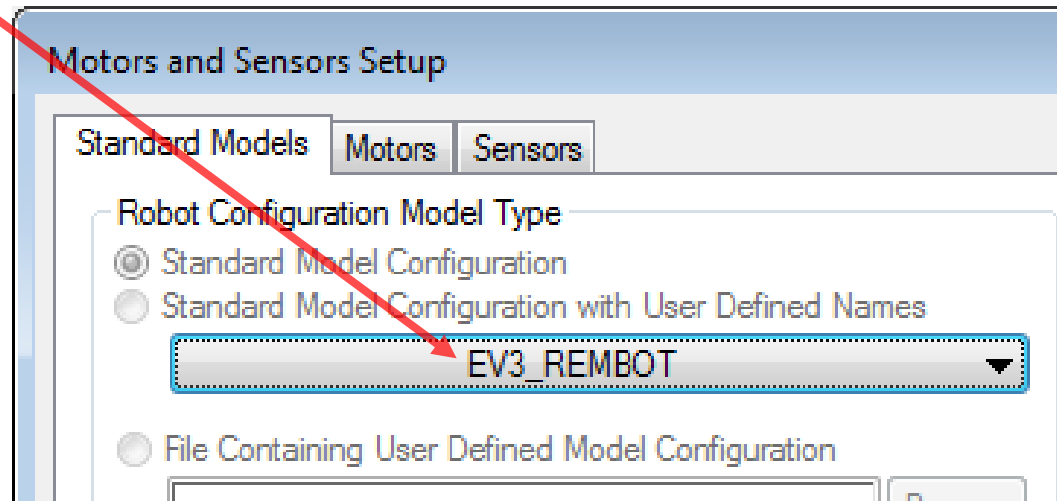
[예제] 장애물 만나면 후진하기

1. 새 파일 생성

[File]-[New]-[New File] 선택

2. 모터 및 센서 설정

- ① [Robot]-[Motor and Sensors Setup] 선택
- ② Standard Models 탭에서 [Select Configuration] 클릭
- ③ [EV3_REMBOT] 선택



[예제] 장애물 만나면 후진하기

3. 소스 코드 작성

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

task main()
{
    while(SensorValue(touchSensor)==0)
    {
        
    }

    setMotorSpeed(leftMotor, -75);
    setMotorSpeed(rightMotor, -75);
    sleep(2000);
}
```

[예제] 장애물 만나면 후진하기

3. 저장

[File]-[Save] 선택

☞ 파일명: touch-backward.c

4. 컴파일 & 다운로드

[Robot]-[Compile and Download Program] 선택

☞ 단축키: <F5>

5. 실행

Program Debug 창에서 [Start] 클릭

초음파 센서 활용



초음파 센서(ultrasonic sensor)는...

- 초음파가 물체에 부딪혀 돌아오는 시간으로 거리를 계산
- 주변 광량이나 물체의 색에 영향을 받지 않음
- 측정 가능한 거리는 3cm이상, 250cm이하
- 일반적으로 4번 입력 포트에 연결

[예제] 장애물 만나면 멈추기

- 조건
 - 75% 출력으로 전진
 - 장애물과의 거리가 20cm가 되면 멈춤
- 모터 및 센서 설정
 - EV3_REMBOT
- 파일명
 - sonar-stop.c

[예제] 장애물 만나면 멈추기

1. 새 파일 생성

[File]-[New]-[New File] 선택

2. 모터 및 센서 설정

- ① [Robot]-[Motor and Sensors Setup] 선택
- ② Standard Models 탭에서 Select Configuration 클릭
- ③ EV3_REMBOT 선택

3. 저장

- ① [File]-[Save] 선택
- ② sonar-stop 입력 후 저장

[예제] 장애물 만나면 멈추기

4. 소스 코드 작성

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

```
task main()
```

```
{
```

```
    int distance = 20;
```

```
    while(SensorValue[ 센서 이름 ] > distance)
```

```
    // 초음파 센서와 전방 장애물까지의 거리가 20cm 이상이면 전지하는 반복문
```

```
    {
```

```
        setMotorSpeed(motorB, 75);
```

```
        setMotorSpeed(motorC, 75);
```

```
    }
```

```
    setMotorSpeed(motorB, 0);
```

```
    setMotorSpeed(motorC, 0);
```

```
}
```

SensorValue의 경우 소괄호와 대괄호 둘 다 사용 가능

leftMotor, rightMotor 사용 가능

[예제] 장애물 만나면 멈추기

5. 컴파일 & 다운로드

[Robot]-[Compile and Download Program] 선택

☞ 단축키 <F5>

6. 실행

Program Debug 창에서 Start 클릭

[예제] 움직이는 장애물 따라가기

- 조건
 - 장애물과의 거리가 항상 30cm 유지
- 모터 및 센서 설정
 - EV3_REMBOT
- 파일명
 - sonar-follow.c

[예제] 움직이는 장애물 따라가기

1. 새 파일 생성

[File]-[New]-[New File] 선택

2. 모터 및 센서 설정

- ① [Robot]-[Motor and Sensors Setup] 선택
- ② Standard Models 탭에서 Select Configuration 클릭
- ③ EV3_REMBOT 선택

3. 저장

- ① [File]-[Save] 선택
- ② 파일명에 sonar-follow 입력

[예제] 움직이는 장애물 따라가기

4. 코드 작성

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

task main()
{
    const int distanceToMaintain = 30;
    int currentDistance = 0;

    while(true)
    {
        currentDistance = SensorValue[sonarSensor];

        displayCenteredBigTextLine(4, "Dist: %3d cm", currentDistance);

        if ((distanceToMaintain - currentDistance) < -2) {
            motor[motorB] = 25;
            motor[motorC] = 25;
        }

        else if ((distanceToMaintain - currentDistance) > 2) {
            motor[motorB] = -25;
            motor[motorC] = -25;
        }

        else{
            motor[motorB] = 0;
            motor[motorC] = 0;
        }

        sleep(50);
    }
}
```

[예제] 움직이는 장애물 따라가기

5. 컴파일 & 다운로드

[Robot]-[Compile and Download Program] 선택

☞ 단축키 <F5>

6. 실행

Program Debug 창에서 [Start] 클릭

[예제] 움직이는 장애물 따라가기

• 코드 설명

```
:
const int distanceToMaintain = 30;
//장애물과 유지하고 싶은 거리를 30cm로 지정

int currentDistance = 0;
//거리 값 저장 변수 선언 및 0으로 초기화

while(true)
{
    currentDistance = SensorValue[sonarSensor];
    //초음파 센서로 측정한 장애물과의 거리를 변수에 저장

    displayCenteredBigTextLine(4, "Dist: %3d cm", currentDistance);
    //장애물과의 거리를 브릭 화면의 다섯 번째 줄의 중앙에 표시
:

```

<출력 형식 지정자>

- %: 형식 지정자 시작
- 3: 출력 폭을 3칸으로 지정
- d: 10진 정수로 표시



[예제] 움직이는 장애물 따라가기

- 코드 설명

currentDistance > 32

//

:

```
if ((distanceToMaintain - currentDistance) < -2) {  
    motor[motorB] = 25;  
    motor[motorC] = 25;  
}
```

//장애물과의 거리가 32cm보다 커지면 25% 파워로 전진

```
else if ((distanceToMaintain - currentDistance) > 2) {  
    motor[motorB] = -25;  
    motor[motorC] = -25;  
}
```

//장애물과의 거리가 28cm보다 작아지면 25% 파워로 후진

```
else{  
    motor[motorB] = 0;  
    motor[motorC] = 0;  
}
```

// 장애물과의 거리가 28cm에서 32cm까지의 범위에 있으면 정지

:

컬러 센서 활용



컬러 센서(Color Sensor)는...

- 사물의 밝기, 사물의 색상, 주변의 밝기를 측정
- 일반적으로 3번 입력 포트에 연결

컬러 센서 모드

- 반사광 모드(reflected mode)
 - 컬러 센서의 기본 모드
 - 사물의 명도(밝기)를 측정
 - ☞ 빨간색 LED 조명을 쏘아 반사되어 돌아오는 빛의 양을 측정
 - 값의 범위: 0~100
 - ☞ 클수록 밝은 색



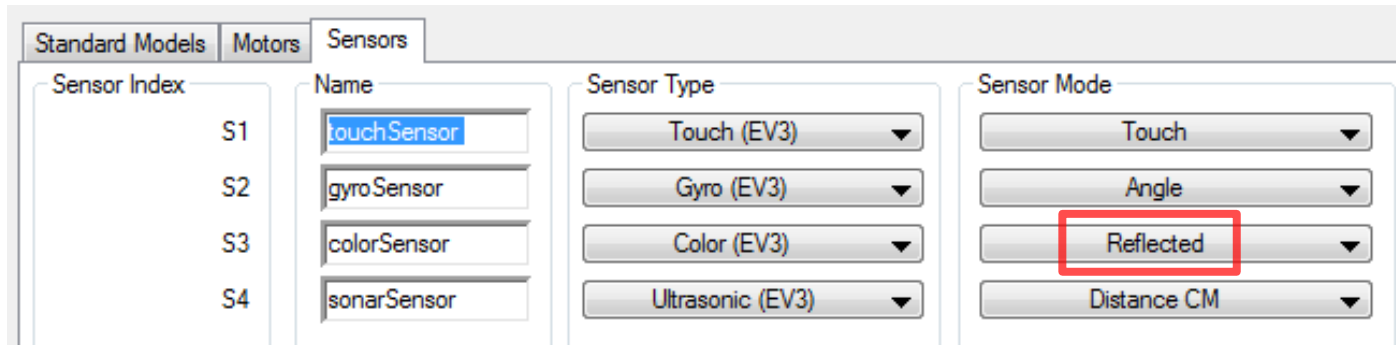
Reflected Light Intensity Mode

컬러 센서 모드

- 반사광 모드(reflected mode)

- 설정

- ① [Robot]-[Motor and Sensors Setup] 선택
- ② Sensors 탭 선택
- ③ Sensor Mode 항목에서 **Reflected** 선택



EV3_REMBOT의 기본 값은 Color 모드

컬러 센서 모드

- 컬러 모드(color mode)

- 7개의 색을 측정

색		값
색 없음(None)		0
검정(Black)		1
파랑(Blue)		2
초록(Green)		3
노랑(Yellow)		4
빨강(Red)		5
흰색(White)		6
갈색(Brown)		7

0은 어떤 물체도 감지되지 않았음을 의미

컬러 센서 모드

- 컬러 모드(color mode)

- 설정

- ① [Robot]-[Motor and Sensors Setup] 선택
- ② Sensors 탭 선택
- ③ Sensor Mode 항목에서 Color 선택

Standard Models				Motors		Sensors	
Sensor Index		Name	Sensor Type		Sensor Mode		
S1			No Sensor		Not Applicable		
S2			No Sensor		Not Applicable		
S3		colorSensor	Color (EV3)		Color		
S4			No Sensor		Not Applicable		

컬러 센서 모드

- 주변광 모드(ambient mode)

- 주변의 밝기 측정
- 값의 범위: 0~100
 - ☞ 클수록 주변이 밝음



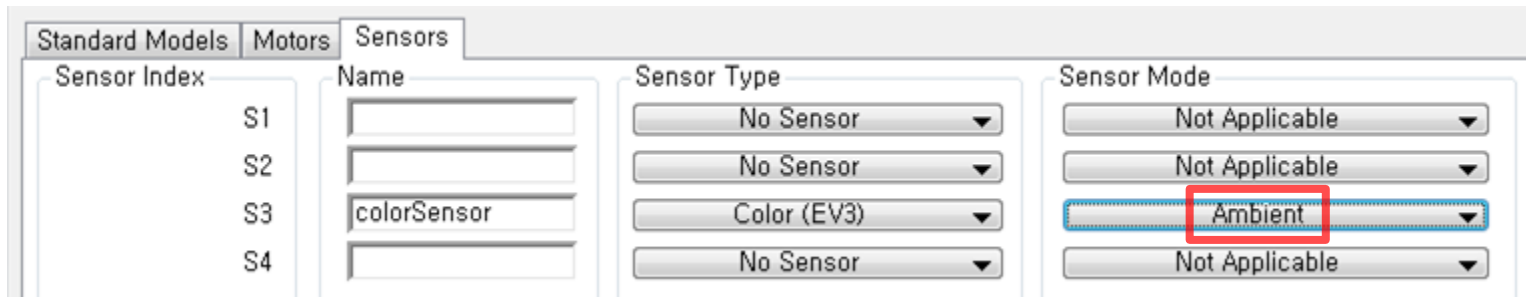
Ambient Light Intensity Mode

컬러 센서 모드

- 주변광 모드(ambient mode)

- 설정

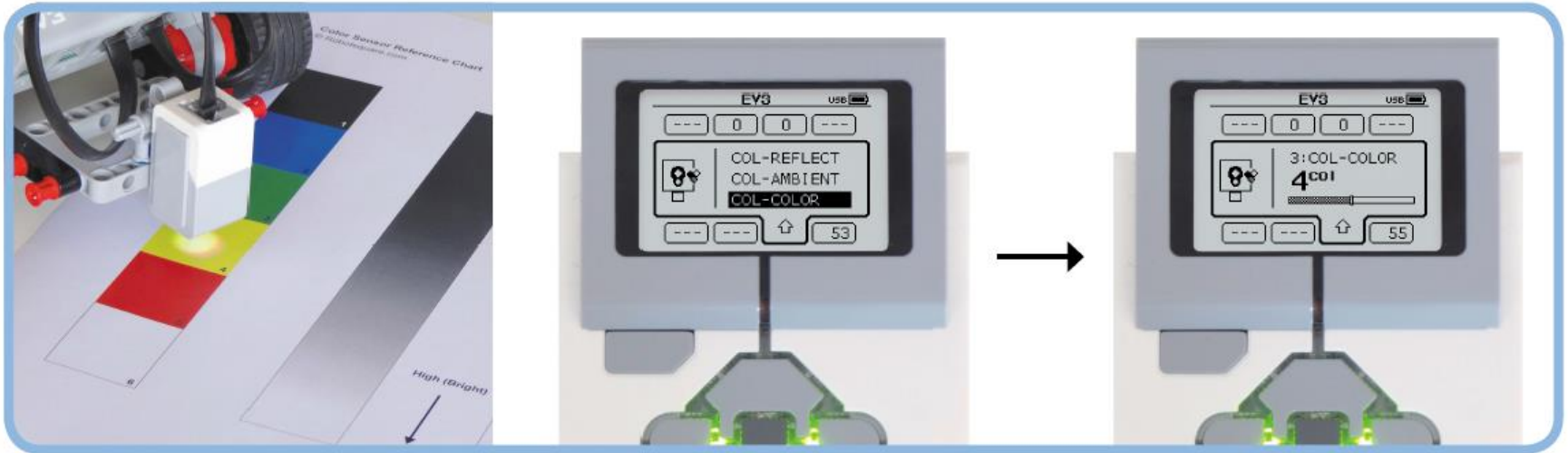
- ① [Robot]-[Motor and Sensors Setup]
- ② Sensors 탭 선택
- ③ Sensor Mode 항목에서 Ambient 선택



컬러 모드의 측정 값 확인

• 컬러 센서 측정 값 보기

- ① EV3 컨트롤러의 브릭 앱 탭에서 Port View 선택
- ② '오른쪽' 버튼 두 번 눌러 3번 입력 포트에 이동
- ③ '가운데' 버튼 누른 후 COL-COLOR 선택



반사광 모드의 분겟값

- 검은색과 흰색의 분겟값(threshold value)이란
검은색과 흰색의 분리 기준이 되는 값

- 분겟값 구하는 방법

- ① 검은색 반사광 측정
- ② 흰색 반사광 측정
- ③ (검은색 측정값 + 흰색 측정값) ÷ 2

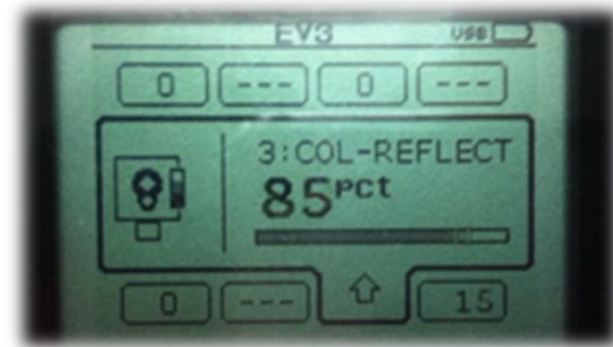
측정값은...

센서의 상태, 조명의 상태, 로봇의 속도 등에
의해 영향을 받을 수 있음



(퀴즈)

- 검은색의 반사광이 4, 흰색의 반사광이 85일 경우 분깃값은?



44 또는 45

[예제] 검은색 라인에서 정지하기

- 조건

- 반사된 빛의 밝기가 분깃값보다 크면 전진
- 반사된 빛의 밝기가 분깃값보다 작거나 같으면 정지

- 모터 및 센서 설정

- Standard Model EV3_REMBOT 선택
- 컬러 센서 모드 **Reflected** 선택

- 사용 함수

- SensorValue(센서 이름)
- setMotorSpeed(모터 이름, 속도)

[예제] 검은색 라인에서 정지하기

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

```
task main()
```

```
{
```

```
    while(SensorValue[colorSensor] > 44) {
```

컬러 센서의 측정 값이 44보다 크면 50% 파워로 전진

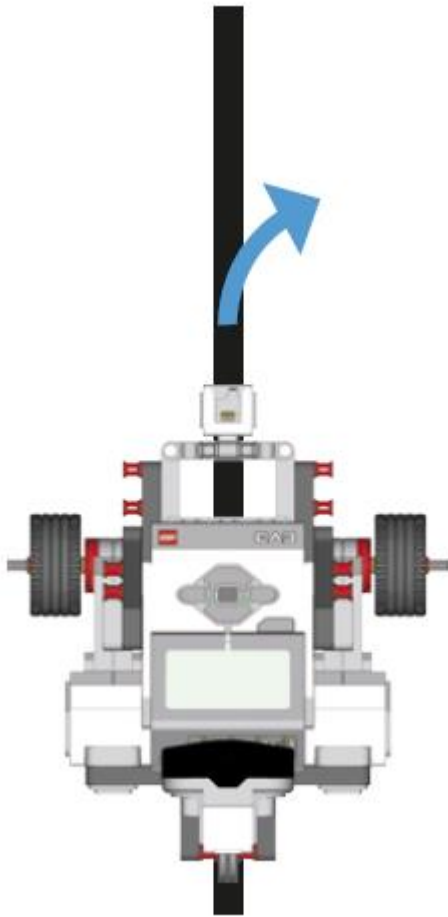
```
        setMotorSpeed(motorB, 50);  
        setMotorSpeed(motorC, 50);
```

```
    }  
  
    setMotorSpeed(motorB, 0);
```

```
    setMotorSpeed(motorC, 0);
```

```
}
```

라인 트레이싱 원리



검은색 인식



흰색 인식

[예제] 검은색 라인 따라가기

- 조건

- 검은색 라인의 왼쪽 부분을 따라 전진
- 장애물에 닿으면 멈춤

- 모터 및 센서 설정

- Standard Model EV3_REMBOT 선택
- 컬러 센서 모드 **Reflected** 선택

- 사용 함수

- SensorValue(센서 이름)
- setMotorSpeed(모터 이름, 속도)

[예제] 검은색 라인 따라가기

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

```
task main()  
{
```

```
    while(SensorValue(colorSensor) < 44) {
```

장애물에 닿지 않으면 라인 트레이싱 반복

```
        if(SensorValue(colorSensor) < 44) {  
            setMotorSpeed(motorB, 20);  
            setMotorSpeed(motorC, 60);  
            turnLeft();  
        }
```

검은색 감지하면 왼쪽으로 턴

```
        else {  
            setMotorSpeed(motorB, 60);  
            setMotorSpeed(motorC, 20);  
            turnRight();  
        }
```

흰색 감지하면 오른쪽으로 턴

```
    setMotorSpeed(motorB, 0);  
    setMotorSpeed(motorC, 0);
```

```
}
```

[예제] 색이름 표시하기

- 조건
 - 감지된 색상의 이름을 브릭 화면에 표시
- 모터 및 센서 설정
 - Standard Model EV3_REMBOT 선택
 - 컬러 센서 모드 Color 선택
- 사용 제어문 및 함수
 - switch() {}
 - SensorValue()
 - displayCenteredBigTextLine()

[예제] 색이름 표시하기

```
task main()
{
    short currentColor;

    while(1) {
        currentColor = SensorValue[colorSensor];

        switch(currentColor) {
            case 0: displayCenteredBigTextLine(7, "None"); break;
            case 1: displayCenteredBigTextLine(7, "Black"); break;
            case 2: displayCenteredBigTextLine(7, "Blue"); break;
            case 3: displayCenteredBigTextLine(7, "Green"); break;
            case 4: displayCenteredBigTextLine(7, "Yellow"); break;
            case 5: displayCenteredBigTextLine(7, "Red"); break;
            case 6: displayCenteredBigTextLine(7, "White"); break;
            case 7: displayCenteredBigTextLine(7, "Brown"); break;
            default: displayCenteredBigTextLine(7, "Unknown");
        }

        sleep(20);
    }
}
```

측정된 컬러 값에 따라 다른 문장 실행

switch문 실행을 종료하는 분기문

1초에 50번 색 감지

05 센서 활용

1. 터치, 초음파, 컬러 센서 활용하기
2. 엔코더 제어하고 활용하기

[탐구] 36cm 전진

B4 용지의 긴 쪽 길이(36cm)만큼
전진 후 멈추는 코드를 작성하시오.



엔코더 제어

- 엔코더(encoder)란
 - 모터에 내장된 회전 각도 측정 센서
- 엔코더 제어의 필요성
 - 속도나 시간에 상관 없이 원하는 거리를 이동시킬 수 있음

엔코더 제어를 하지 않을 경우,
원하는 거리를 이동시키기 위한 속도나 시간을 찾기가 어려움!

setMotorSpeed() 값

sleep() 값

엔코더 제어

- EV3를 원하는 거리만큼 이동시키는 방법

원하는 이동 거리가 X_{cm} 이면..

- ① X_{cm} 이동하기 위한 바퀴의 회전 각도 Y° 를 계산
- ② 바퀴의 회전 각도가 Y° 가 될 때까지 전진

?

바퀴의 회전 각도 = 모터의 엔코더 값

이동 거리에 따른 엔코더 값 계산

이동시킬 거리에 해당하는
바퀴의 회전 각도(엔코더 값)를 계산하기 위해서는
바퀴의 둘레 길이를 알아야 한다

왜냐하면,
바퀴의 둘레 길이는 회
전 각도가 360° 일 때의
이동 거리이므로...

바퀴의 둘레 길이를 계산하기 위해서는...
바퀴의 반지름을 알아야 한다



EV3 바퀴의 반지름
2.8cm

이동 거리에 따른 엔코더 값 계산

- 바퀴의 둘레 길이 구하기

$$2\pi r = 2 \times 3.14 \times 2.8 = 17.58 \text{ (cm)}$$

즉,
바퀴가 한 바퀴(360°) 굴러갈 때 이동 한 거리는...



17.58cm

이동 거리에 따른 엔코더 값 계산

- Xcm만큼 이동시키기 위한 회전 각도 Y° 구하기 (생각1)

17.58cm를 이동시키기 위해 360° 회전이 필요하므로...

$$17.58\text{cm} : 360^\circ = X\text{cm} : Y^\circ$$

따라서...

$$\underset{\text{(엔코더 값)}}{Y} = \frac{360 \times X}{17.58}$$

이동 거리에 따른 엔코더 값 계산

- Xcm만큼 이동시키기 위한 회전 각도 Y° 구하기 (생각2)

바퀴 둘레가 17.58cm 일 때...

$$Y^\circ \text{ 회전한 바퀴의 이동 거리 } X = 17.58 \times \frac{Y}{360}$$

따라서...

$$X\text{cm 이동한 바퀴의 회전 각도 } Y = \frac{360 \times X}{17.58}$$

(퀴즈)

- 로봇을 36cm 이동시키기 위한 회전 각도(엔코더 값)는?

Ycm만큼 이동시키는 회전 각도는...

$$\frac{360 \times Y}{17.58}$$

따라서...

$$\frac{360 \times 36}{17.58} = 737 (^{\circ})$$

[예제] 엔코더 값을 이용한 전진1

- 조건

- 이동 거리로 엔코더 값을 구하는 함수 정의
- 정의한 함수로 36cm를 이동하는 엔코더 값을 구하여 변수에 저장
- 측정한 현재 엔코더 값이 저장된 엔코더 값보다 작으면 전진 반복

- 사용 함수

- resetMotorEncoder(모터 이름)
 - ☞ 모터의 엔코더 값 초기화
- getMotorEncoder(모터 이름)
 - ☞ 모터의 엔코더 값 반환

[예제] 엔코더 값을 이용한 전진1

```
int convert(int dist)
{
    return 360 * dist / 17.58;
}
```

이동 거리로 엔코더 값을 구하는 함수

```
task main()
{
```

```
    int enc_degree;
    enc_degree = convert(36);
```

함수에서 리턴된 엔코더 값 저장

```
    resetMotorEncoder(motorA);
```

왼쪽 모터의 엔코더 값 초기화

```
    while(getMotorEncoder(motorB) < enc_degree) {
        setMotorSpeed(motorB, 20);
        setMotorSpeed(motorC, 20);
    }
```

왼쪽 모터의 현재 엔코더 값이 목표 엔코드 값보다 작으면 전진 반복

```
    setMotorSpeed(motorB, 0);
    setMotorSpeed(motorC, 0);
```

양쪽 모터 멈춤

```
}
```

[예제] 엔코더 값을 이용한 전진2

- 조건

- 양쪽 모터 엔코더 값 초기화
- 양쪽 모터 목표 엔코더 값 1871, 속도 75 설정
- 한쪽 모터 멈출 때까지 현재 상태 유지

- 모터 및 센서 설정

- motorB: lm
- motorC: rm

[예제] 엔코더 값을 이용한 전진2

- 주요 함수

- resetMotorEncoder(모터)

- ☞ 모터의 엔코더 값 초기화

- setMotorTarget(모터, 목표 엔코더 값, 속도)

- ☞ 모터의 목표 엔코더 값 및 속도 설정

- waitUntilMotorStop(모터)

- ☞ 모터가 멈출 때까지 대기

setMotorTarget와 waitUntilMotorStop을 사용하면
반복문을 사용하지 않아도
목표 엔코더 값까지 이동시킬 수 있음

[예제] 엔코더 값을 이용한 전진2

```
#pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
#pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
```

```
task main()
```

```
{
```

```
resetMotorEncoder(lm);
```

양쪽 모터 엔코더 값 초기화

```
resetMotorEncoder(rm);
```

```
setMotorTarget(lm, 737, 20);
```

양쪽 모터 목표 엔코더 값 737, 속도 20 설정

```
setMotorTarget(rm, 737, 20);
```

```
waitUntilMotor
```

한쪽 모터 멈출 때까지 현재 상태 유지

```
}
```

