



로봇활용 SW교육 지침서

The NEXT ROBOT with EV3

EV3로 배우는 블록 코딩 & C언어

2017년 2학기

인제대학교 헬스케어IT 학과

이상훈



education

나눔바른고딕

HandsOn
Technology

Weekly plan (2nd semester, 2017)

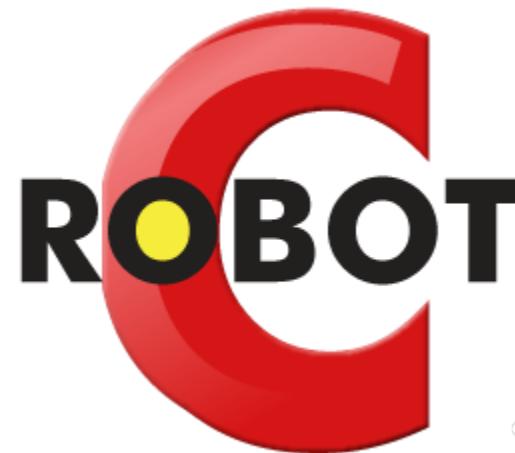
- **wk01 : Introduction to curriculum & current state of HW-SW coding**
- **wk02 : LME blocking coding-1: Start & How To**
- **wk03 : LME blocking coding-2: Loop & Driving**
- **wk04 : LME blocking coding-3: Project 1. driving base**
- **wk05 : LME blocking coding-4: Sensors**
- **wk06 : ,wk11 : Special talk by prof. Redwoods Yi**
- **wk07 :**
- **wk08 : Mid-term Exam.**
- **wk09 : LME blocking coding-5: Math and Logic**
- **wk10 : LME blocking coding-6: Summary & Project**
- **wk11 : Special talk by CEO of HandsOn Tech.**
- **wk12 : RobotC coding-1: Intro**
- **wk13 : RobotC coding-2: function, array, motor**
- **wk14 :**
- **wk15 : Final exam.**





wk13: RobotC II.

LEGO® Mindstorms® EV3
powered by LEGO® MINDSTORMS® Education



창의공학교육의 멘토



education

HandsOn
Technology

1부 EV3로 배우는 블록 코딩

I. LEGO® MINDSTORMS® Education EV3

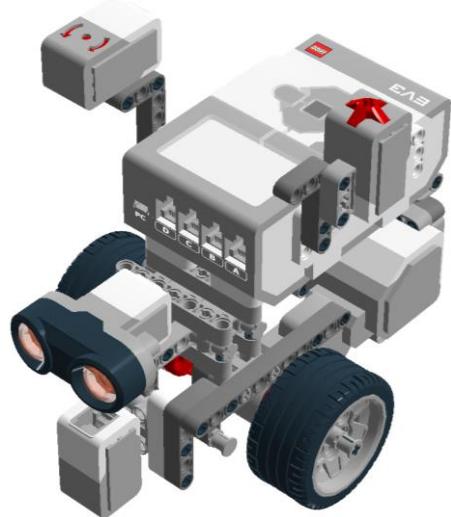
1. EV3와 NXT 비교, 브릭 인터페이스
2. Starting block coding

- ✓ Awake EV3!
- ✓ Loop & Driving
- ✓ Driving base
- ✓ Sensors
- ✓ Advanced coding



2. ROBOTC 기초

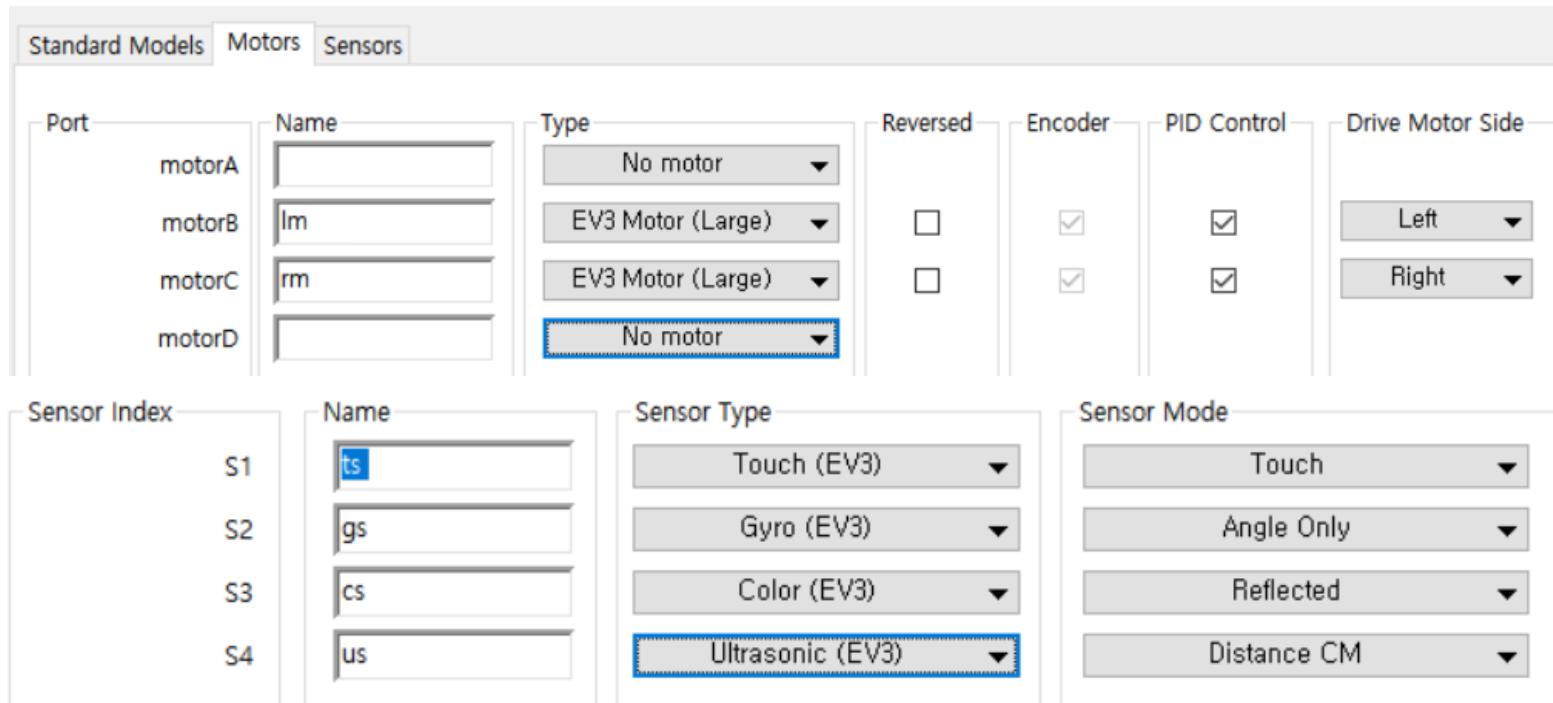
- ROBOTC 규칙
- 시간(지연) 함수
- 데이터 저장 및 계산
- 조건 선택/실행
- 반복 실행
- 함수
- 배열



모터 및 센서 설정

- ‘ROBOTC 기초’의 소스 코드 작성을 위해 모터 및 센서를 설정해야 함
 - 단, 일부 코드는 센서 설정의 변경이 필요함

Motors and Sensors Setup



데이터 저장 및 계산

- 변수 : 원하는 데이터를 저장하기 위한 메모리 내부의 공간
 - 연산을 위해 기존 데이터를 재사용해야 하는 경우 때문에 변수가 필요함
- 변수명 작성 규칙
 - 대·소문자와 숫자 그리고 밑줄문자(_)를 사용하며, 대소문자를 구별함
 - 숫자로 시작할 수 없으며 기본 예약어와 라이브러리 함수명을 사용할 수 없음

DIY-1. Show team information

- 다음 내용을 포함하는 팀정보를 출력하는 `EVnn_team_info.c`를 작성하시오.
- 각 정보를 변수에 담아서 사용.
 - 팀id,
 - 팀구성원 이름 (영문)
 - 학번 끝 2자리 수
 - 팀 logo (영문)

Hint code

```

4   task main()
5   {
6       //char x = 'a'; // 'a' -> 97
7       //x = 98;
8       // 한글 도움말 입력 가능
9       // 한글 출력은 ????
10      string s1 = "EV00";
11      string s2 = "Redwoods, Chaos";
12      int acNum = 77;
13      string s31 = "Stay hungry, ";
14      string s32 = "stay foolish.";
15
16      displayBigTextLine(0, "Show EV00 Info.");
17      displayBigTextLine(3, "Name: %s", s2);
18      displayBigTextLine(6, "acNUM: %d", acNum);
19      displayBigTextLine(9, s3);
20      sleep(5000);
21
22  }

```

연산자

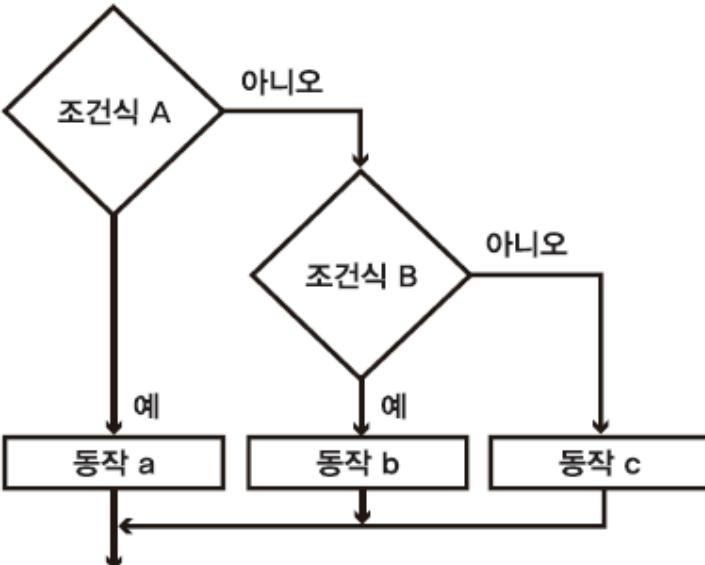
- 연산자의 개념
 - 연산자는 수학 기호라고 생각할 수 있으며 숫자를 계산하고 서로 비교할 때 사용됨
- 산술 연산자 : 사칙연산과 나눗셈에서의 나머지를 표현하는 연산자
 - 프로그래밍에서 가장 기본적인 연산자

연산자	연산식	의미
덧셈	$x + y$	x 와 y 의 합
뺄셈	$x - y$	x 와 y 의 차
곱셈	$x * y$	x 와 y 의 곱
나눗셈	x / y	x 를 y 로 나눌 때의 몫
나머지	$x \% y$	x 를 y 로 나눌 때의 나머지

if~else문의 확장

- ‘조건식 A’를 만족할 경우 ‘동작 a’를 실행하고, 그렇지 않고 ‘조건식 B’를 만족할 경우 ‘동작 b’를 실행함, 그렇지 않으면 ‘동작 c’를 실행

```
if(조건식 A)
{
    동작 a;
}
else if(조건식 B)
{
    동작 b;
}
else
{
    동작 c;
}
```



초음파 센서를 이용한 주행

2-14

초음파 센서를 이용한 주행(if~else문의 확장)

```

01 #pragma config(Sensor, S4, ss, sensorEV3_Ultrasonic)
02 #pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
03 #pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
04
05 task main()
06 {
07     while(1)
08     {
09         if(getUSDistance(ss) < 10)
10         {
11             setMotorSpeed(lm, 30);
12             setMotorSpeed(rm, -30);
13             sleep(1000);
14         }
15         else if(getUSDistance(ss) < 30)
16         {
17             setMotorSpeed(lm, 30);
18             setMotorSpeed(rm, 30);
19         }
20         else
21         {
22             setMotorSpeed(lm, 50);
23             setMotorSpeed(rm, 50);
24         }
25     }
26 }
```

07

무한 반복(07번~25번 줄) 한다. while문은 이후의 장에서 다루도록 한다.

09~14

초음파 센서의 값이 10보다 작을 경우 포인트 턴을 수행한다.

15~19

9번 줄의 if문이 거짓이며(초음파 센서 값이 10보다 크거나 같음), 초음파 센서의 값이 30보다 작을 경우 30의 모터 값으로 전진한다.

20~24

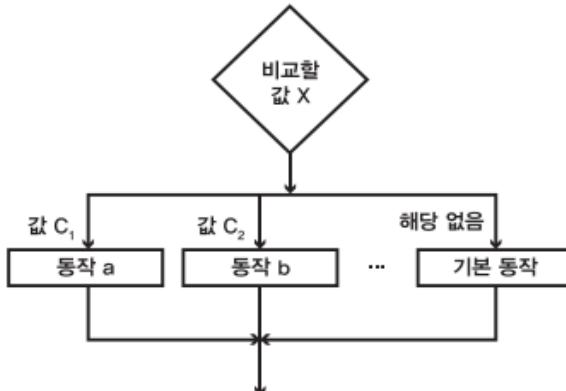
9번 줄과 15번 줄의 if문이 모두 거짓일 경우 50의 모터 값으로 전진한다.

switch~case문

- 비교할 값 X가 C_1 이면 ‘동작 a’를, C_2 이면 ‘동작 b’를 실행하고,
(중략), 모두 아니면 ‘기본 동작’을 실행함 아니면 기본 동작 실행

```
switch(비교할 값 X)
{
    case 값  $C_1$ :
        동작 a;
        break;

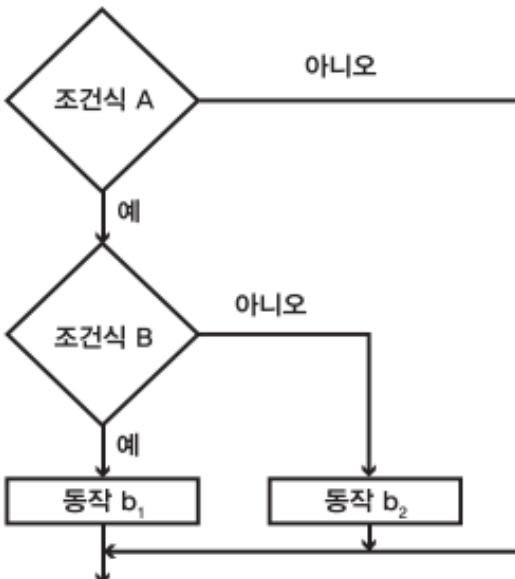
    case 값  $C_2$ :
        동작 b;
        break;
    (중략)
    default:
        기본 동작;
}
```



복합조건 선택/실행

- 조건문을 복합적으로 사용하여 복잡한 문제 상황 제어 가능

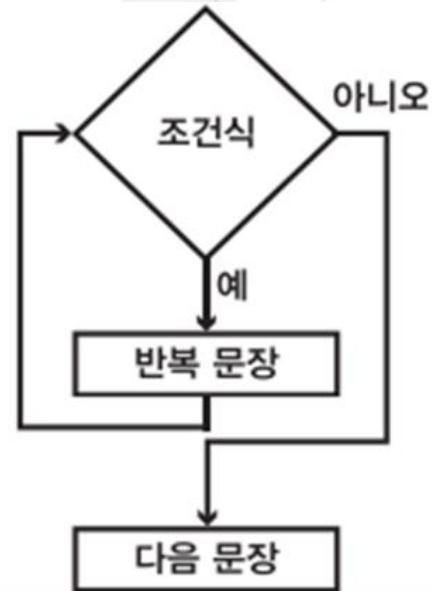
```
if(조건식 A)
{
    if(조건식 B)
    {
        동작 b1;
    }
    else
    {
        동작 b2;
    }
}
```



while문

- ‘조건식’이 참일 경우 ‘명령’을 수행한 후 다시 ‘조건식’을 검사한다.
- ‘조건식’이 거짓일 경우 명령을 수행하지 않고 while문을 종료한다.

```
while( 조건식 )
{ "조건식"이 참일 때 수행할 명령; }
```



do~while문

- 기본적인 개념은 while문과 유사하지만 do~while문의 경우 반복 문장을 적어도 한 번 이상 실행

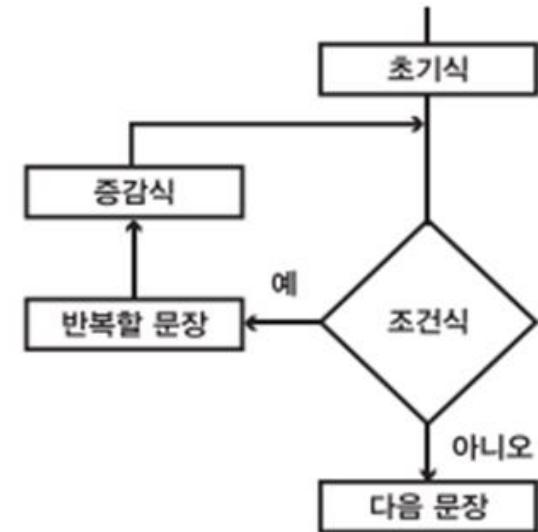
```
do
{ "조건식"이 참일 때 수행할 명령; }
while( 조건식 );
```



for문

- ‘초기식’ 설정 후 ‘조건식’이 참이면 ‘반복할 문장’을 수행하고 ‘증감식’ 수행
‘조건식’이 거짓이면 ‘반복할 문장’을 수행하지 않고 for문 탈출

```
for(초기식; 조건식; 증감식)
{ "조건식"이 참일 때 반복할 문장; }
```



중첩 반복문

- 조건문을 중첩하여 사용하는 복합 제어문처럼 반복문을 중첩하여 사용하는 것을 의미함
- 일반적으로 for, while을 중첩하여 많이 사용함

컬러 센서를 이용한 주행

2-12

컬러 센서를 이용한 주행(if문)

```

01 #pragma config(Sensor, S1, ts, sensorEV3_Touch)
02 #pragma config(Sensor, S3, cs, sensorEV3_Color, modeEV3Color_Ambient)
03 #pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
04 #pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
05
06 #define EDGE_VALUE 30
07
08 task main()
09 {
10     while(getTouchValue(ts) == 0)
11     {
12         if(getColorReflected(cs) < EDGE_VALUE)
13         {
14             setMotorSpeed(lm, -50);
15             setMotorSpeed(rm, 50);
16         }
17         if(getColorReflected(cs) >= EDGE_VALUE)
18         {
19             setMotorSpeed(lm, 50);
20             setMotorSpeed(rm, 50);
21         }
22     }
23     setMotorSpeed(lm, 0);
24     setMotorSpeed(rm, 0);
25 }

```

- 02 컬러 센서를 주변광(ambient) 모드로 설정한다.
- 06 컬러 센서 값의 경계값으로 30을 설정한다. 이는 유동적일 수 있다.
- 10 터치 센서가 눌리지 않았을 경우에 아래 명령을 반복한다. while문은 이후의 장에서 다루도록 한다.
- 12 컬러 센서의 값이 설정한 경계값보다 작으면 하위 명령을 실행한다.
- 14~15 반시계 방향으로 포인트 턴하도록 양쪽 모터 값을 변경한다.
- 17 컬러 센서의 값이 설정한 경계값 이상이면 하위 명령을 실행한다.
- 19~20 50의 모터 값으로 전진한다.
- 23~24 터치 센서가 눌리면 while문을 탈출한 뒤에 정지한다.

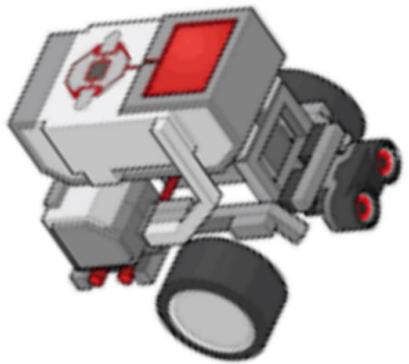


education

HandsOn
Technology

RobotC project-1

- 주행하다가 검은 띠에서 정지 2초 후 계속 주행
- 벽 앞 10cm에서 밀당.



EVnn_MilDang.c
로 저장.



Best code

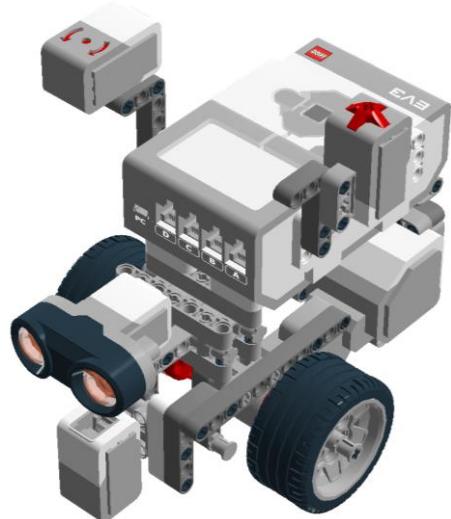
```
#define EDGE_VALUE 30

task main()
{
    while(getColorReflected(cs)>EDGE_VALUE)
    {
        setMotorSpeed(lm,50);
        setMotorSpeed(rm,50);
    }
    if(getColorReflected(cs) <= EDGE_VALUE)
    {
        setMotorSpeed(lm,0);
        setMotorSpeed(rm,0);
        sleep(2000);
    }
    setMotorSpeed(lm,50);
    setMotorSpeed(rm,50);

    while(1) {
        if(getUSDistance(us)<=10) {
            setMotorSpeed(lm,-50);
            setMotorSpeed(rm,-50);
        }
        else {
            setMotorSpeed(lm,50);
            setMotorSpeed(rm,50);
        }
    }
}
```

2. ROBOTC 기초

- ROBOTC 규칙
- 시간(지연) 함수
- 데이터 저장 및 계산
- 조건 선택/실행
- 반복 실행
- **함수**
- **배열**



함수의 개념

- 주어진 입력값에 대한 결과값을 출력하는 관계이자, 명령의 집합

- 프로그래밍 언어에서의 함수의 종류

① 프로그램을 시작하는 main() 함수

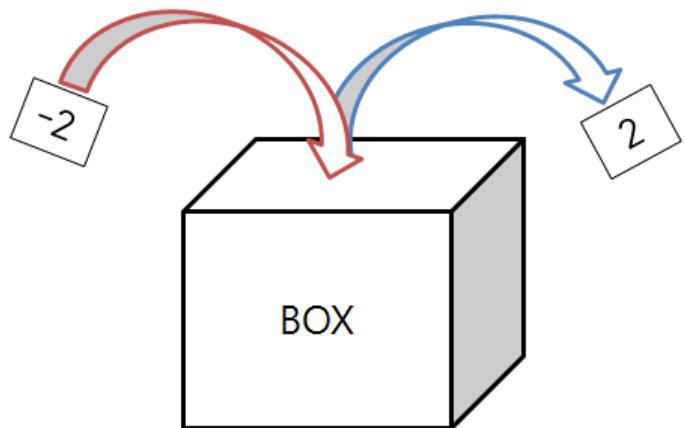
프로그램 실행시 가장 먼저 실행되는 함수

② 프로그래밍 언어에서 제공하는 라이브러리 함수

별도의 정의 없이 활용이 가능

③ 프로그래머가 필요에 의해 정의한 사용자 정의 함수

사용자가 필요에 따라 직접 만든 함수



함수의 개념

◆ 사용자 정의 함수

함수에 필요한 입력 매개변수, 반환값, 함수의 이름, 내용을 다음과 같이 정의

void move(int speed, int time)		
void	move	(int speed, int time)
반환값의 자료형	함수의 이름	입력 매개변수의 자료형과 이름

```
void move(int speed, int time)
{
    setMotorSpeed(lm, speed);
    setMotorSpeed(rm, speed);
    sleep(time);
}
```

함수의 개념

• 사용자 정의 함수 사용

앞서 정의한 함수는 다음과 같이 호출 가능

move(100, 3000)	
move	(100, 3000)
호출할 함수의 이름	함수에 입력할 매개 변수 매개변수의 자료형은 함수와 같아야 한다. 쉼표(.)로 여러 개의 매개 변수를 구분한다.

함수의 개념

- 함수를 사용하면 코드 중 중복된 내용을 정리할 수 있고,
매개 변수의 입력에 따라 다양한 행동 수행 가능
코드의 길이를 줄이고 가독성을 높일 수 있음
- 함수의 원형을 선언한 후, 함수를 정의하면(코드 2-23) 그렇지 않을 경우보다
(코드 2-22) 프로그램의 실행 흐름과 일치하고, 가독성이 뛰어난 코드를 작성
할 수 있음

함수를 이용한 주행

- 함수의 원형의 사용 유무에 따른 비교

2-22	함수를 이용한 주행(함수의 원형 미사용)
01	#pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
02	#pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
03	
04	<u>void move(int speed, int time)</u>
05	{
06	setMotorSpeed(lm, speed);
07	setMotorSpeed(rm, speed);
08	sleep(time);
09	}
10	
11	task main()
12	{
13	move(100, 1000);
14	move(-50, 2000);
15	move(25, 4000);
16	}
줄번호	코드 설명
04~09	매개 변수 speed, time의 값을 전달받아 move() 함수가 실행된다.
13~15	서로 다른 모터 값과 시간을 이용하여 move() 함수를 호출한다.

2-23	함수를 이용한 주행(함수의 원형 사용)
01	#pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
02	#pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
03	
04	<u>void move(int speed, int time);</u>
05	
06	task main()
07	{
08	move(100, 1000);
09	move(-50, 2000);
10	move(25, 4000);
11	}
12	
13	<u>void move(int speed, int time)</u>
14	{
15	setMotorSpeed(lm, speed);
16	setMotorSpeed(rm, speed);
17	sleep(time);
18	}
줄번호	코드 설명
04	move() 함수의 정보를 미리 정의한다. 이것을 함수의 원형(prototype)이라고 한다.
08~10	서로 다른 모터 값과 시간으로 로봇을 움직인다.
13~18	move() 함수의 내용을 나중에 정의한다.

함수의 설계 - 반환 값

- ◆ 반환 값이 없는 함수

반환 값이 없는 **void**형 함수는 보통 입력 받은 매개 변수에 따라 작업을 수행하는 함수로 쓰임

- ◆ 반환 값이 있는 함수

반환 값이 정수 혹은 실수형인 함수는 작업을 수행하고 얻은 결과를 반환한다.

```
return a;
```

함수의 설계

2-24

함수의 반환값을 이용한 주행(int형 함수)

```

01 #pragma config(Sensor, S3, cs, sensorEV3_Color)
02 #pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
03 #pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
04
05 int search(int speed, int time);
06
07 task main()
08 {
09     int LightValue;
10     LightValue = search(50, 1000);
11 }
12
13 int search(int speed, int time)
14 {
15     setMotorSpeed(lm, speed);
16     setMotorSpeed(rm, speed);
17     sleep(time);
18
19     return getColorReflected(cs);
20 }
```

05 int형 함수 search() 함수의 원형을 선언한다.

09 int형 변수 LightValue를 선언한다.

10 변수 LightValue에 search() 함수의 반환값을 대입한다.

13~20 로봇은 time 시간동안 움직인 후 컬러 센서 cs의 센서값을 반환한다.

함수의 설계

- ◆ 자기 자신을 호출하는 재귀함수
 - 재귀함수란 return문을 이용하여 호출한 함수가 자기 자신인 함수를 말함
 - 재귀함수에서는 return 코드를 이용하여 자기 자신을 호출
 - 호출이 멈추는 부분을 포함해야 재귀함수의 무한 호출이 일어나지 않음

n번째 피보나치 수를 계산하는 함수

2-25	n번째 피보나치 수를 반환하는 재귀 함수
01	<u>int fibo(int n)</u>
02	{
03	if(n <= 1) return n;
04	<u>return fibo(n-2) + fibo(n-1);</u>
05	}
06	task main()
07	{
08	int n = 10;
09	displayTextLine(1, "fibo(%d) = %d", n, <u>fibo(n)</u>);
10	sleep(5000);
11	}
줄번호	코드 설명
01	n번째 피보나치 수열을 구하는 fibo() 함수를 정의한다.
03	n = 0, 1일 때 각각 0과 1의 값을 반환한다.
04	n0 2 이상일 때, fibo(n-2)와 fibo(n-1)을 호출하여 두 반환값의 합을 반환한다.
08~10	n(10)번째 피보나치 수를 구하여 디스플레이에 5초간 나타낸다.

배열의 선언과 이용

- 배열 : 동일한 자료형의 데이터들을 여러 개 저장할 수 있는 데이터 저장 장소, 여러 개의 값이 하나의 이름을 공유하기 때문에 자료 조작이 편리함
- 배열의 이름, 배열 요소의 자료형, 배열의 크기를 이용한 선언 필요

int data[100]		
int	data	[100]
배열의 자료형	배열의 이름	배열의 크기(0~99)

배열의 선언과 이용

- 배열 안에 저장된 각각의 데이터들에 접근하기 위해서 인덱스를 이용하여 원하는 자료의 배열 내에서의 위치를 지정할 필요가 있음
- **인덱스(index, 첨자)** : 배열 내에서의 번호, **항상 0부터 시작**되며 정수 값으로 배열의 크기 이상의 값을 가질 수 없음. 상수뿐만 아니라 변수로도 지정 가능
- 인덱스로 변수를 지정하면 반복문을 통해 변수에 변화를 줘 배열의 자료들에 간단히 접근하고 처리하는 것이 가능함

배열의 선언과 이용

2-26	1차원 배열의 선언과 초기화
01	task main()
02	{
03	int data[5];
04	
05	<u>data[0] = 1;</u>
06	data[4] = 5;
07	
08	displayBigTextLine(1, "%d %d", data[0], data[4]);
09	sleep(5000);
10	}
줄번호	코드 설명
03	정수형 변수 5개를 저장할 수 있는 배열 data[]를 선언한다.
05	data[]의 첫 번째 저장 공간에 1을 저장한다.
06	data[]의 5번째 저장 공간에 5를 저장한다.
08~09	디스플레이의 1번째 줄에 data[0], data[4]의 값을 5초간 나타낸다.



배열의 초기화

- main() 함수의 외부에서 전역 배열로 선언하면?
 - 배열도 일종의 변수(집합)이므로 전역 배열과 지역 배열이 존재함
 - 배열의 모든 원소들의 값이 자동으로 0으로 초기화됨
 - 전역 배열이므로 모든 함수에서 읽고 쓰는 것이 가능함

- 반복문을 이용하여 초기화하면?
 - 모든 배열 공간을 0이 아닌 다른 수로도 쉽게 초기화시킬 수 있음
 - 전역 배열 보다는 주로 지역 배열에서의 초기화에 이용

1차원 배열의 초기화

2-27	반복문을 이용한 1차원 배열의 초기화
01	task main()
02	{
03	int data[5];
04	int i;
05	
06	for(i = 0; i < 5; i++)
07	{
08	data[i] = 0;
09	}
10	
11	for(i = 0; i < 5; i++)
12	{
13	displayTextLine(1, "data[%d] = %d", i, data[i]);
14	sleep(2000);
15	}
16	}
줄번호	코드 설명
06~09	반복문을 이용하여 data[0]부터 data[4]까지의 값을 모두 0으로 초기화시킨다.
11~15	디스플레이의 1번째 줄에 data[0]부터 data[4]까지의 값을 각각 2초간 나타낸다.



2차원 배열

- 보다 많은 요소들을 한 번에 다루어야 할 경우 2차원 배열을 이용
(ROBOTC는 3차원 이상의 배열을 지원하지 않는다.)
- **int matrix [4] [4];**
- 첫 번째 인덱스는 행(row) 번호, 두 번째 인덱스는 열(column) 번호라고 함

행 \ 열	0	1	2	3
0				
1				
2				
3				

2차원 배열의 초기화

2-28

반복문을 이용한 2차원 배열의 초기화

```

01 task main()
02 {
03     int data[5][5];
04     int data1[5][5];
05     int i, j;
06
07     for(i = 0; i < 5; i++)
08         for(j = 0; j < 5; j++)
09             data[i][j] = 0;
10
11     for(i = 0; i < 5; i++)
12         for(j = 0; j < 5; j++)
13             data1[i][j] = i + j;
14

```

03~04

5 * 5 크기의 2차원 배열 2개를 선언한다.

07~09

반복문을 이용하여 배열 data[][]를 모두 0으로 초기화시킨다.

11~13

반복문을 이용하여 data1[][]을 아래 그림과 같이 초기화시킨다.

열	0	1	2	3	4
행	0	1	2	3	4
1	1	2	3	4	5
2	2	3	4	5	6
3	3	4	5	6	7
4	4	5	6	7	8

2차원 배열의 초기화 후 출력

2-28	반복문을 이용한 2차원 배열의 초기화	
15	for(i = 0; i < 5; i++)	
16	for(j = 0; j < 5; j++)	
17	displayStringAt((j+1)*10, 100-(i+1)*10, "%d ", data[i][j]);	15~18
18	sleep(5000);	디스플레이에 배열 data[][]의 모든 데이터를 5초간 나타낸다.
19		
20	for(i = 0; i < 5; i++)	
21	for(j = 0; j < 5; j++)	
22	displayStringAt((j+1)*10, 100-(i+1)*10, "%d ", data1[i][j]);	20~23
23	sleep(5000);	디스플레이에 배열 data1[][]의 모든 데이터를 5초간 나타낸다.
24	}	

Tip

displayStringAt() 명령어

- LCD 디스플레이의 특정 좌표에 문자열 출력 가능
- 픽셀 한 칸을 단위로 (x,y) 좌표를 설정할 수 있으며 좌표 평면의 원점은 LCD 디스플레이의 왼쪽 하단이다.

displayStringAt((j+1)*10, 100-(i+1)*10, "%d ", data[i][j])				
displayStringAt()	(j+1)*10	100-(i+1)*10	“%d ”	data[i][j]
지정 좌표에 문자열을 출력한다.	문자열의 x 좌표	문자열의 y 좌표	문자열의 형식지정자	문자열

DIY-1. Show 2-dimensional array

- 아래에 정의된 이차원 배열 map을 출력하는 EVnn_2darray.c를 작성하시오.

2-29

2차원 배열의 선언과 초기화

```

01 task main()
02 {
03     int i, j, map[5][5] = {
04         {0, 0, 1, 1, 1},
05         {1, 0, 1, 0, 0},
06         {1, 1, 0, 1, 0},
07         {0, 0, 0, 0, 1},
08         {1, 1, 1, 1, 0}
09     };
10
11     for(i = 0; i < 5; i++)
12         for(j = 0; j < 5; j++)
13             displayStringAt( (j+1)*10, 100-(i+1)*10, "%d ", map[i][j] );
14     sleep(5000);
15 }
```

03~09

5 * 5 크기의 2차원 배열 map[][]을 선언하고 아래 그림과 같이 초기화시킨다.

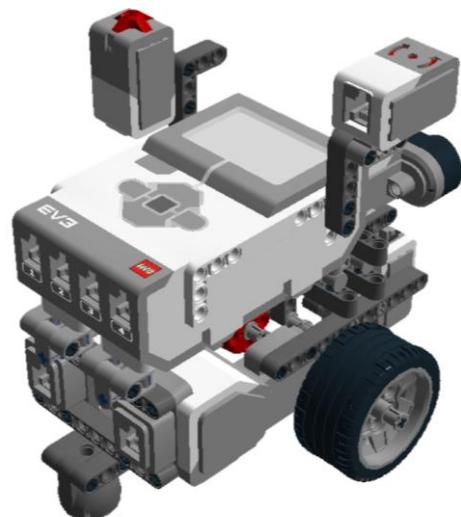
행 \ 열	0	1	2	3	4
0	0	0	1	1	1
1	1	0	1	0	0
2	1	1	0	1	0
3	0	0	0	0	1
4	1	1	1	1	0

11~14

디스플레이에 배열 map[][]의 모든 데이터를 5초간 나타낸다.

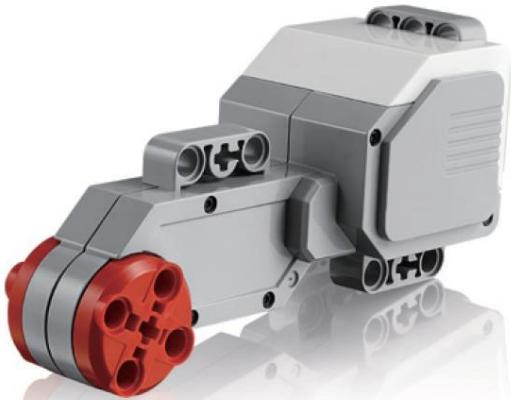
3. 액츄에이터 제어

- ◆ 모터
- ◆ 디스플레이
- ◆ 스피커
- ◆ LED



모터

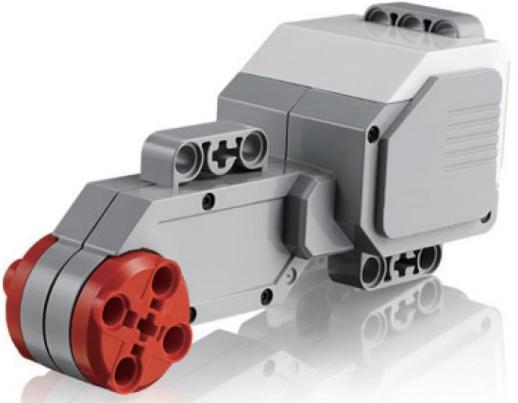
- -100~100 사이의 정수로 모터의 출력량을 지정 가능
- 절댓값이 커질수록 모터 출력 세기가 증가
- 값이 음수일 때는 역회전, 양수일 때는 정회전을 하며 0일 경우 정지





EV3와 NXT 모터

- EV3의 라지 모터 사양이 NXT의 서보 모터와 동일하기 때문에 EV3와 NXT 간 모터 호환 가능
- EV3 라지 모터는 보다 빠르고 복잡한 조립을 위해 크기와 모양이 최적화 됨



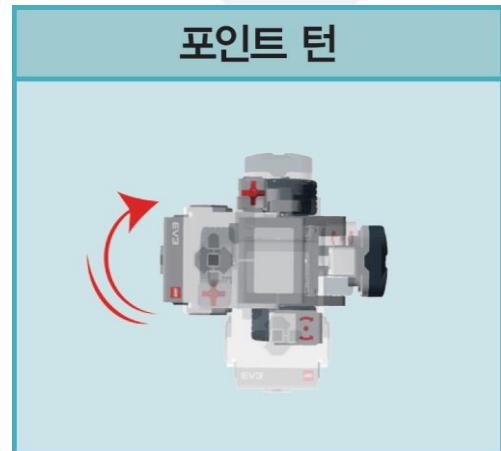
전진/후진/정지

3-1	모터의 전진, 후진 및 정지
01	#pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
02	#pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
03	
04	task main()
05	{
06	setMotorSpeed(lm, 50);
07	setMotorSpeed(rm, 50);
08	sleep(3000);
09	
10	setMotorSpeed(lm, -50);
11	setMotorSpeed(rm, -50);
12	sleep(3000);
13	
14	setMotorSpeed(lm, 0);
15	setMotorSpeed(rm, 0);
16	}
줄번호	코드 설명
06~08	3초간 전진한다.
10~12	3초간 후진한다.
14~15	정지한다.

로봇의 회전

- ▶ **포인트 턴** : 같은 모터 값을 한쪽에는 정방향으로, 다른 쪽은 역방향으로 지정하여 제자리에서 회전

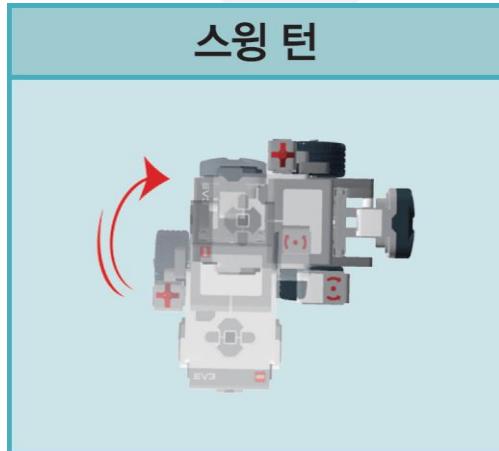
3-2	로봇의 포인트 턴
01	#pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
02	#pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
03	
04	task main()
05	{
06	setMotorSpeed(lm, 50);
07	setMotorSpeed(rm, -50);
08	sleep(5000);
09	
10	setMotorSpeed(lm, 0);
11	setMotorSpeed(rm, 0);
12	}
줄번호	코드 설명
06~08	5초간 오른쪽으로 포인트 턴한다.
10~11	정지한다.



로봇의 회전

- **스윙 턴** : 한쪽 모터는 정지하고 다른 쪽만 회전시켜 바퀴 사이의 거리를 반지름으로 하는 원 모양으로 회전

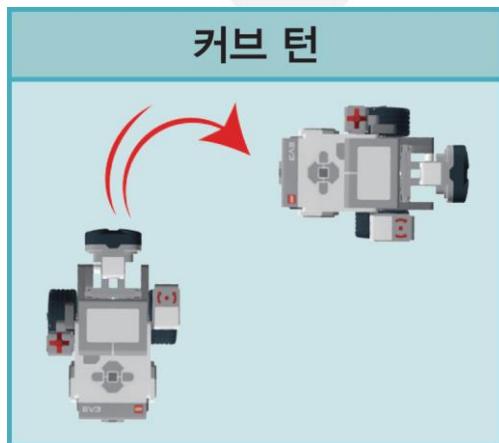
3-3	로봇의 스윙 턴
01	#pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
02	#pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
03	
04	task main()
05	{
06	<u>setMotorSpeed(lm, 50);</u>
07	<u>setMotorSpeed(rm, 0);</u>
08	sleep(5000);
09	
10	setMotorSpeed(lm, 0);
11	setMotorSpeed(rm, 0);
12	}
줄번호	코드 설명
06~08	5초간 오른쪽으로 스윙 턴한다.
10~11	정지한다.



로봇의 회전

- **커브 턴** : 두 모터의 모터 값을 다르게 해 모터 값이 더 작은 방향으로 큰 원을 그리며 회전

3-4	로봇의 커브 턴
01	#pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
02	#pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
03	
04	task main()
05	{
06	setMotorSpeed(lm, 50);
07	setMotorSpeed(rm, 20);
08	sleep(5000);
09	
10	setMotorSpeed(lm, 0);
11	setMotorSpeed(rm, 0);
12	}
줄번호	코드 설명
06~08	5초간 오른쪽으로 커브 턴한다.
10~11	정지한다.



디스플레이

- EV3 브릭의 LCD 디스플레이 장치에 다양한 형태의 데이터 출력 가능
- 로봇의 동작 중에도 센서 값, 변수값의 변화 확인 가능 - 테스트, 디버깅에 유용
- 사용자와의 인터페이스 용도로 사용 가능

문자열 출력

- 문자열은 문장의 시작과 끝 위치에 큰 따옴표(" ")를 넣어 표현

3-5	문자열 값의 출력
01	task main()
02	{
03	displayBigTextLine(2, <u>"Hello,"</u>);
04	displayBigTextLine(4, <u>"EV3 is fun!"</u>);
05	sleep(5000);
06	}
줄번호	코드 설명
03~05	LCD 디스플레이의 2번 줄에 "Hello,"를, 4번 줄에 "EV3 is fun!"을 5초간 나타낸다.

형식 지정자

- 변수값을 표현하기 위해 변수의 자료형에 해당하는 형식 지정자 사용

자료형	형식지정자	의미
int	%d	정수형 자료
char	%c	문자형 자료
float	%f	실수형 자료
string	%s	문자열형 자료

줄번호	초음파 센서 값의 출력
01	#pragma config(Sensor, S4, ss, sensorEV3_Ultrasonic)
02	
03	task main()
04	{
05	while(1)
06	{
07	displayBigTextLine(2,"DISTANCE = <u>%dcm</u> ", getUSDistance(ss));
08	sleep(500);
09	}
10	}
줄번호	코드 설명
05~09	LCD 디스플레이의 2번 줄에 초음파 센서의 값을 0.5초마다 바꾸어가며 나타낸다.

선 출력

- 선의 시작 지점과 종료 지점 좌표를 지정하여 직선 출력
- x 좌표 : 0~178, y 좌표 : 0~128까지 설정 가능 (원점 : 디스플레이 좌측 하단)

3-7	초음파 센서 값의 그래프 출력
01	#pragma config(Sensor, S4, ss, sensorEV3_Ultrasonic)
02	
03	task main()
04	{
05	int dist, pos_x = 1;
06	while(1)
07	{
08	if(pos_x > 178)
09	{
10	eraseDisplay();
11	pos_x = 1;
12	}
13	dist = getUSDistance(ss) / 3;
14	<u>drawLine(pos_x, 0, pos_x, dist);</u>
15	sleep(200);
16	pos_x += 2;
17	}
18	}

08~12 LCD 디스플레이의 오른쪽 끝까지 도달하면 화면의 내용을 모두 지우고, pos_x의 값을 1로 초기화한다.

13 그래프가 화면에 전부 표시될 수 있도록 초음파 센서의 값을 3으로 나눈 뒤 dist 변수에 대입한다.

14~15 dist 변수의 값을 세로 길이로 하는 선을 0.2초 간격으로 그린다.

16 pos_x의 값을 2 증가시켜 다음 선을 그릴 위치로 한다.

ROBOTC

a C Programming Language for Robotics

drawLine

- Draws a line between two points, specified by sets of X and Y coordinates

Code Sample

```
//Displays two diagonal lines on the EV3 LCD Screen
//which form an "X" shape
drawLine(10, 10, 100, 100);
drawLine(10, 100, 100, 10);
```

void drawLine(const int xPos, const int yPos, const int xPosTo, const int yPosTo)

Parameter	Explanation	Data Type
Return Type	The function returns no value.	void
xPos	The X-coordinate of the first point.	int
yPos	The Y-coordinate of the first point.	int
xPosTo	The X-coordinate of the second point.	int
yPosTo	The Y-coordinate of the second point.	int



음계 출력

음계	옥타브	5	6
도(C)		523	1047
도#(C#)		554	1109
레(D)		587	1175
레#(D#)		622	1245
미(E)		659	1319
파(F)		699	1397
파#(F#)		740	1475
솔(G)		784	1568
솔#(G#)		831	1661
라(A)		880	1760
라#(A#)		932	1865
시(B)		988	1976

↳ 일반적으로 많이 사용하는 음계에 대한 주파수

줄번호	동요 '나비야' 연주
01	task main()
02	{
03	int frequency[7] = {523, 587, 659, 699, 784, 880, 988};
04	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};
05	int i, t;
06	for(i = 0; i < 27; i++)
07	{
08	t = song[i];
09	playImmediateTone(frequency[t], 50);
10	if(i == 2 i == 5 i == 12 i == 19) sleep(1000);
11	else sleep(500);
12	}
13	}
줄번호	코드 설명
03	각 음계의 주파수 값을 배열에 순서대로 넣어준다.
04	'나비야' 노래의 계이름 순서를 넣어준다.
08	표현할 음을 변수 t에 넣는다.
09	변수 t의 값에 해당하는 주파수를 표현한다.
10~11	조건에 따라 0.5초 또는 1초간 음을 유지한다.

ROBOTC

a C Programming Language for Robotics

playImmediateTone

- Immediately play tone at frequency & duration ahead of queued requests.

Code Sample

```
// Play tone according to light sensor readings
// Wait 200 milliseconds before checking again

PlayImmediateTone(440, 50);

wait1Msec(200);
```

void playImmediateTone(int frequency, int durationIn10MsecTicks)

Parameter	Explanation	Data Type
Return Type	The function returns no value	void
frequency	The frequency of the tone to play	int
durationIn10MsecTicks	How long to play the tone (measured in units of 10 milliseconds)	int



내부 사운드 파일 출력

내부 사운드	
soundBeepBeep	soundLowBuzz
soundBlip	soundLowBuzzShort
soundDownwardTones	soundShortBlip
soundException	soundUpwardTones
soundFastUpwardTones	

내부 사운드 파일 연주	
3~9	<pre> 01 task main() 02 { 03 playSound(soundBeepBeep); 04 sleep(2000); 05 playSound(soundException); 06 sleep(2000); 07 }</pre>
코드 설명	
03~04	'soundBeepBeep' 소리를 2초간 출력한다.
05~06	'soundException' 소리를 2초간 출력한다.

ROBOTC

a C Programming Language for Robotics

playSound

- Play one of the system predefined sounds
- Predefined Sounds:
- soundBlip
- soundBeepBeep
- soundDownwardTones
- soundUpwardTones
- soundLowBuzz
- soundFastUpwardTones
- soundShortBlip
- soundException
- soundLowBuzzShort

Code Sample

```
// Play the sound, 'soundBeepBeep'
playSound(soundBeepBeep);
```

void playSound(TSounds sound)

Parameter	Explanation	Data Type
Return Type	The function returns no value.	void
sound	Sound to play	TSounds



외부 사운드 출력

- ROBOTC 4.X 자체에서 제공하지 않는 소리는 외부 파일을 불러와 출력 가능
(.rsf 형식 지원)

3-10	외부 사운드 파일 연주
01	task main()
02	{
03	<u>playSoundFile("Bravo");</u>
04	sleep(2000);
05	}
줄번호	코드 설명
03~04	'Bravo' 소리를 2초간 출력한다.

LED 출력

Color number	동작	Color number	동작
0	ledOff	5	ledRedFlash
1	ledGreen	6	ledOrangeFlash
2	ledRed	7	ledGreenPulse
3	ledOrange	8	ledRedPulse
4	ledGreenFlash	9	ledOrangePulse

줄번호	10가지 패턴으로 LED 켜기
01	task main()
02	{
03	int col;
04	for(col = 0; col < 10; col++)
05	{
06	<u>setLEDColor(col);</u>
07	sleep(3000);
08	<u>setLEDColor(ledOff);</u>
09	sleep(1000);
10	}
11	setLEDColor(ledOff);
12	}
줄번호	코드 설명
06~07	여러 가지 색상으로 LED를 3초간 On 한다.
08~09	LED를 1초간 Off 한다.

4. 센서 활용

- 터치 센서
- 컬러 센서
- 초음파 센서
- 자이로 센서
- 엔코더 제어
- 타이머 활용
- 데이터 로깅



Tip

EV3와 NXT 센서 간 호환성

1. EV3 센서를 NXT 지능형 브릭과 함께 사용할 수 있나요?

→ 불가능하다. EV3 센서는 디지털 방식이며, 따라서 NXT 지능형 브릭과 함께 사용할 수 없다.

2. EV3 센서를 NXT 센서와 함께 사용할 수 있나요?

→ 가능하다.

3. NXT 센서를 EV3 브릭과 함께 사용할 수 있나요?

→ 가능하다.

터치 센서

- 눌러지고 있는 동안은 1을, 떼어져 있는 동안 0을 계속 반환

터치 센서를 이용한 전진과 후진	
줄번호	코드 설명
01	#pragma config(Sensor, S1, ts, sensorEV3_Touch)
02	#pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
03	#pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
04	
05	task main()
06	{
07	int speed = 30;
08	while(1)
09	{
10	setMotorSpeed(lm, speed);
11	setMotorSpeed(rm, speed);
12	<u>while(getTouchValue(ts) == 0){ }</u>
13	<u>while(getTouchValue(ts) == 1){ }</u>
14	speed *= -1;
15	}
16	}
12	터치 센서가 눌릴 때까지 아무 것도 하지 않고 현재 상태를 유지한다.
13	터치 센서가 떼어졌다가 떼어질 때까지 현재 상태를 유지한다.
14	현재 속도의 값에 -1을 곱한다. (곱할 때마다 정회전과 역회전을 반복한다.)

터치 센서



눌리지 않음



눌림



눌렸다 놓음

- 터치 센서가 눌릴 때 동작함(눌리지 않는 동안 대기함)

`while(getTouchValue(ts) == 0) {};` 이용

- 위의 while문 다음에 로봇의 동작 지시 → 누른 순간 동작

- 터치 센서를 눌렀다 뗐을 때 동작함(시작 버튼 등의 제작 시)

`while(getTouchValue(ts) == 0) {}`

`while(getTouchValue(ts) == 1) {}` : 2개의 while문 이용

- 터치 센서가 1을 연속적으로 반환하는 것 방지

터치 센서

4-2	터치 센서를 이용한 숫자 맞추기 게임
01	#pragma config(Sensor, S1, ts, sensorEV3_Touch)
02	
03	task main()
04	{
05	int num = 0;
06	displayTextLine(1, "When you are ready,");
07	displayTextLine(2, "press the touch sensor.");
08	while(getTouchValue(ts) == 0){ }
09	while(getTouchValue(ts) == 1)
10	{
11	num++;
12	eraseDisplay();
13	displayTextLine(4, "%d", num);
14	sleep(100);
15	}
16	if(num == 50)
17	{
18	eraseDisplay();
19	displayTextLine(2, "WIN!");
20	sleep(5000);
21	}
22	}

터치 센서를 이용한 숫자 맞추기 게임입니다. 터치 센서가 눌리면 게임이 시작되고, 숫자를 맞추면 WIN!이라는 메시지가 출력됩니다.

터치 센서는 EV3 모터 드라이브에 내장된 디지털 타입의 접촉식 센서입니다. 터치 센서가 누르면 1을, 누르지 않으면 0을 반환합니다.

터치 센서를 누르면 게임이 시작됩니다. 터치 센서를 누를 때까지 대기합니다. 터치 센서를 누르고 있는 동안 숫자를 1씩 증가시킵니다. 10초 간격으로 디스플레이에 출력합니다. 숫자를 맞추면 WIN!이라는 메시지가 출력됩니다.

컬러 센서

- 3가지 모드로 설정 가능, 흰색에 가까울수록 센서 값 증가

센서 모드	모드 값	설명
Reflected	0 (기본값)	반사광 모드. 발광 다이오드에서 발생하는 붉은 빛이 면에 반사되어 들어오는 양을 측정하여 0~100까지의 값으로 반환한다.
Ambient	1	주변광 모드. 주변 빛의 양을 측정하여 0~100까지의 값으로 반환한다.
Color	2	컬러 모드. 0(색상 없음)~7(갈색) 중 색상 값을 반환한다.

- 컬러 모드로 설정할 경우 8가지의 색상 값 리턴

색상 값	색	색상 값	색
0	색상 없음	4	노랑
1	검정	5	빨강
2	파랑	6	흰색
3	초록	7	갈색



컬러 센서

4-3 컬러 센서를 이용한 사운드 출력	
줄번호	코드 설명
01 02 03 04 05 06 07 08 09 10 11 12 13 14	#pragma config(Sensor, S3, cs, sensorEV3_Color) #pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder) #pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder) task main() { int col; while(1) { col = <u>getColorReflected(cs);</u> playImmediateTone(col * 10, 50); sleep(500); } }

컬러 센서 → sound & motor

```

1 #pragma config(Sensor, S3, cs, sensorEV3_Color)
2 #pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, driveLeft, encoder)
3 #pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, driveRight, encoder)
4 /*!!Code automatically generated by 'ROBOTC' configuration wizard!!*/
5
6 task main()
7 {
8     int col;
9     while(1) {
10         col = getColorReflected(cs);
11         playImmediateTone(col*10, 50);
12         sleep(500);
13
14         setMotorSpeed(lm, col);
15         setMotorSpeed(rm, col);
16         //sleep(500);
17     }
18
19     setMotorSpeed(lm, 0);
20     setMotorSpeed(rm, 0);
21
22 }
```



초음파 센서

- 표면이 거칠고 단단하며, 크기가 큰 물체까지의 거리를 측정
 - 초음파 센서의 고주파가 물체까지 도달한 시간을 거리로 환산하는 방식
- 표면이 매끄럽거나 굽은 물체는 잘 감지하지 못함
 - 초음파 센서의 고주파가 옷감, 수건 등 미세한 구멍이 많은 물체를 뚫고 지나감



초음파 센서

4-4	초음파 센서를 이용한 장애물 회피
01	#pragma config(Sensor, S4, ss, sensorEV3_Ultrasonic)
02	#pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
03	#pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
04	
05	void go(int speed_l, int speed_r)
06	{
07	setMotorSpeed(lm, speed_l);
08	setMotorSpeed(rm, speed_r);
09	}
10	
11	task main()
12	{
13	int dist;
14	while(1)
15	{
16	dist = getUSDistance(ss);
17	if(dist < 10)
18	{
19	go(-30, -30);
20	sleep(1000);
21	go(0, 0);
22	go(30, -30);
23	sleep(700);
24	go(0, 0);
25	}
26	else go(30, 30);
27	}
28	}

05~09

왼쪽 모터와 오른쪽 모터의 출력값을 입력받아 주행한다.

16

초음파 센서 값을 변수 dist에 저장한다.

17~25

변수 dist의 값이 10 미만이면 후진, 회전하여 회피한다.

26

그렇지 않으면 전진한다.

초음파 센서

4-4

초음파 센서를 이용한 장애물 회피

```

01 #pragma config(Sensor, S4, ss, sensorEV3_Ultrasonic)
02 #pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
03 #pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
04
05 void go(int speed_l, int speed_r)
06 {
07     setMotorSpeed(lm, speed_l);
08     setMotorSpeed(rm, speed_r);
09 }
10
11 task main()
12 {
13     int dist;
14     while(1)
15     {
16         dist = getUSDistance(ss);
17         if(dist < 10)
18         {
19             go(-30, -30);
20             sleep(1000);
21             go(0, 0);
22             go(30, -30);
23             sleep(700);
24             go(0, 0);
25         }
26         else go(30, 30);
27     }
28 }
```

[DIY]

함수 원형을 정의해서 주행하도록 코드를 변경하시오.

→ [Evnn_us_function.c](#)

자이로 센서

- 자이로 센서는 단일 회전축을 중심으로 하는 동작만 감지
- 시계 방향으로 회전하면 센서의 측정 값 증가
- 시계 반대 방향으로 회전하면 센서의 측정 값 감소
- 자이로 센서를 초기화하는 명령이 없음
→ 현재 값을 기준으로 차이 값을 계산해야 함

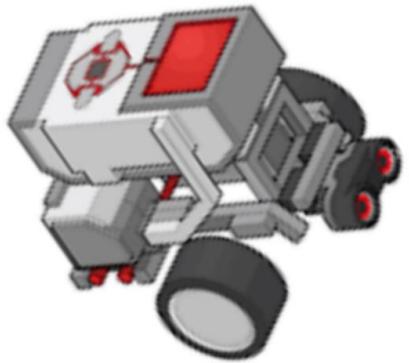


자이로 센서를 이용한 무한 직진

4-5 자이로 센서를 이용한 무한 직진	
줄번호	코드 설명
01	#pragma config(Sensor, S2, gs, sensorEV3_Gyro)
02	#pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl,encoder)
03	#pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl,encoder)
04	task main()
05	{
06	int speed_l, speed_r, curval;
07	int initval = getGyroDegrees(gs);
08	while(1){
09	curval = <u>getGyroDegrees(gs) - initval</u> ;
10	speed_l = 15 - curval;
11	speed_r = 15 + curval;
12	setMotorSpeed(lm, speed_l);
13	setMotorSpeed(rm, speed_r);
14	}
15	}

RobotC project-2

- 자이로센서로 주행하다가 검은 띠에서 정지 2초 후 계속 주행
- 벽 앞 15cm에서 밀당.



EVnn_MilDang2.c
로 저장.



Report

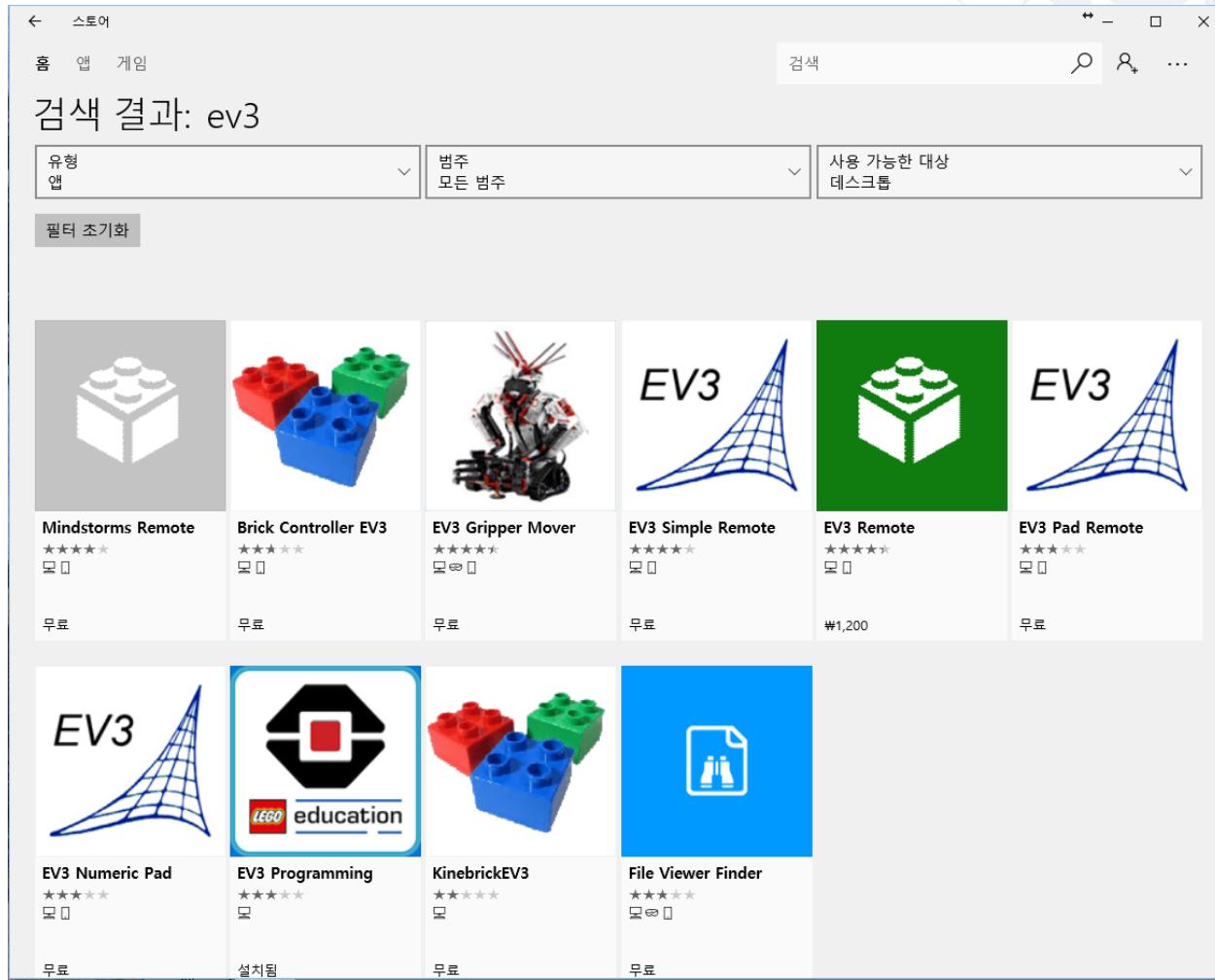
제출파일명 : wk13_EVnn.zip

- 압축할 파일들

- ① EVnn_2darray.c
- ② Evnn_us_function.c
- ③ EVnn_MilDang2.c

Email : **chaos21c@gmail.com**

EV3 Programming App (windows 10)



education

창의공학교육의 멘토

HandsOn
Technology

EV3 Programming App (windows 10)

The screenshot shows the Windows Start menu with the 'EV3 Programming' app icon selected. The app window displays the following information:

- EV3 Programming** by LEGO Education
- ★★★☆☆ (Rating)
- 설명 (Description):

EV3 프로그래밍 언어는 LEGO® Education의 공식 프로그래밍 앱입니다. 직관적인 아이콘 기반의 환경을 사용하는 EV3 프로그래밍 앱은 LEGO MINDSTORMS® Education EV3를 쉽고 효과적으로 시작할 수 있도록 해 줍니다. 이 프로그래밍 앱은 물리적 EV3 로봇과 결합하여 교실 안팎에서 학생들에게 몰입감과 동기를 부여하는 데 필요한 모든 도구를 제공합니다.
- 제공 플랫폼:
 - PC (Icon: monitor)
- 스크린샷:
 - Thumbnail 1: EV3 robot arm with a red sensor.
 - Thumbnail 2: Screenshot of the software interface showing a robot model.
 - Thumbnail 3: Color-coded programming blocks.



education

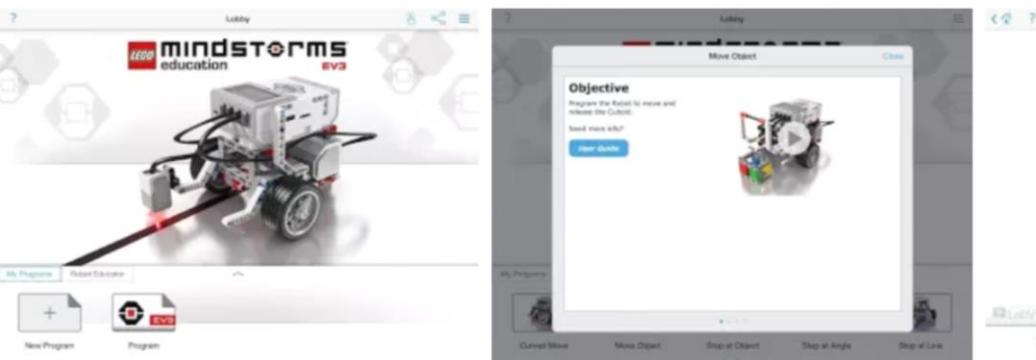
창의공학교육의 멘토

HandsOn
Technology

EV3 Programming App (Android)



LEGO® MINDSTORMS...
LEGO Education
3
4.2 ★ (164 ⚠) • 1만 ↓



LEGO® MINDSTORMS® Education EV3
프로그래밍 언어

[추가 정보](#) [설치](#)

EV3 Programming App (iPad/iPhone)

The image shows a screenshot of an app store page for the "LEGO® MINDSTORMS® Education EV3 Programming" app by LEGO Education. The page includes the app icon, title, developer information, rating, an "OPEN" button, and tabs for "Details", "Reviews", and "Related". Below the store page, there are two photographs illustrating the app's use in a classroom setting.

LEGO® MINDSTORMS® Education EV3 4+
Programming
LEGO Education >
★★★★★ (6)

OPEN

Details Reviews Related

iPad

Inspire students to learn STEM skills.



Develop creativity and critical thinking skills with best-in-class robotics.





로봇활용 SW교육 지침서

The NEXT ROBOT with EV3

EV3로 배우는 C언어와 알고리즘

정웅열 · 최웅선 · 정종광 · 전준호 · 배상용 · 전현석
이선경 · 경다은 · 김제현 · 오범석 · 이찬호 지음

Partnership



education

LEGO education Partner
- Oct. 2011



NATIONAL INSTRUMENTS
OFFICIAL ALLIANCE MEMBER
- Mar. 2003



PITSCO Education
Distributor in Korea
- Jan. 2010



창의공학교육의 멘토
HandsON
Technology