



로봇활용 SW교육 지침서

The NEXT ROBOT with EV3

EV3로 배우는 블록 코딩 & C언어

2017년 2학기

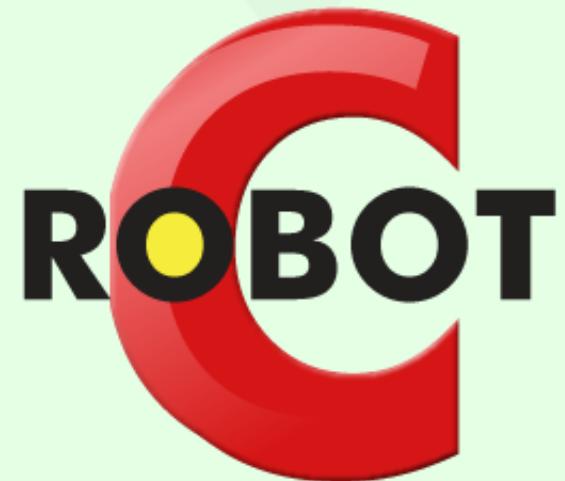
인제대학교 헬스케어IT 학과

이상훈



Weekly plan (2nd semester, 2017)

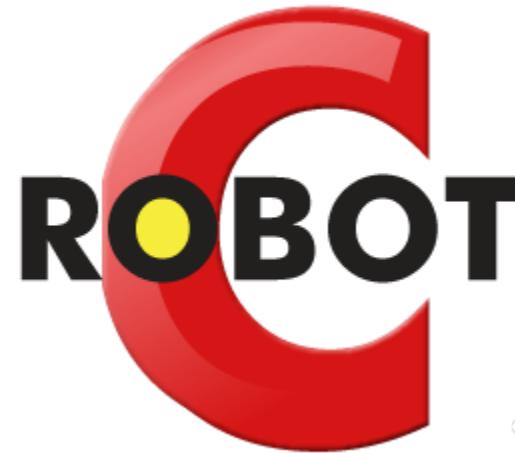
- **wk01 : Introduction to curriculum & current state of HW-SW coding**
- **wk02 : LME blocking coding-1: Start & How To**
- **wk03 : LME blocking coding-2: Loop & Driving**
- **wk04 : LME blocking coding-3: Project 1. driving base**
- **wk05 : LME blocking coding-4: Sensors**
- **wk06 : ,wk11 : Special 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 :**
- **wk14 :**
- **wk15 : Final exam.**





wk12: RobotC

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



창의공학교육의 멘토



education

HandsOn
Technology

1부 EV3로 배우는 블록 코딩

I. LEGO® MINDSTORMS® Education EV3

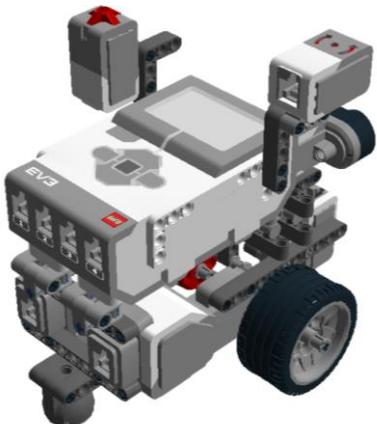
1. EV3와 NXT 비교, 브릭 인터페이스
2. Starting block coding

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



2부. RobotC언어 프로그래밍

1. ROBOTC 개발환경
2. ROBOTC 기초
3. 액츄에이터 제어
4. 센서 활용



www.robotc.net

ROBOTC
a C Programming Language for Robotics

Change Region USA

ROBOTC is a cross-robotics-platform programming language for popular educational robotics systems.

ROBOTC is the premiere robotics programming language for educational robotics and competitions. ROBOTC is a C-Based Programming Language with an Easy-to-Use Development Environment.

[Free Trial Download](#)

Download your evaluation copy today!

- Blazing fast High Performance Firmware.
- Interactive Run-Time Debugger

Don't forget to check out our free training, curriculum, and support!

[ROBOTC Forums](#) [Natural Language Programming](#) [Robot Virtual Worlds](#) [Robot Store](#)



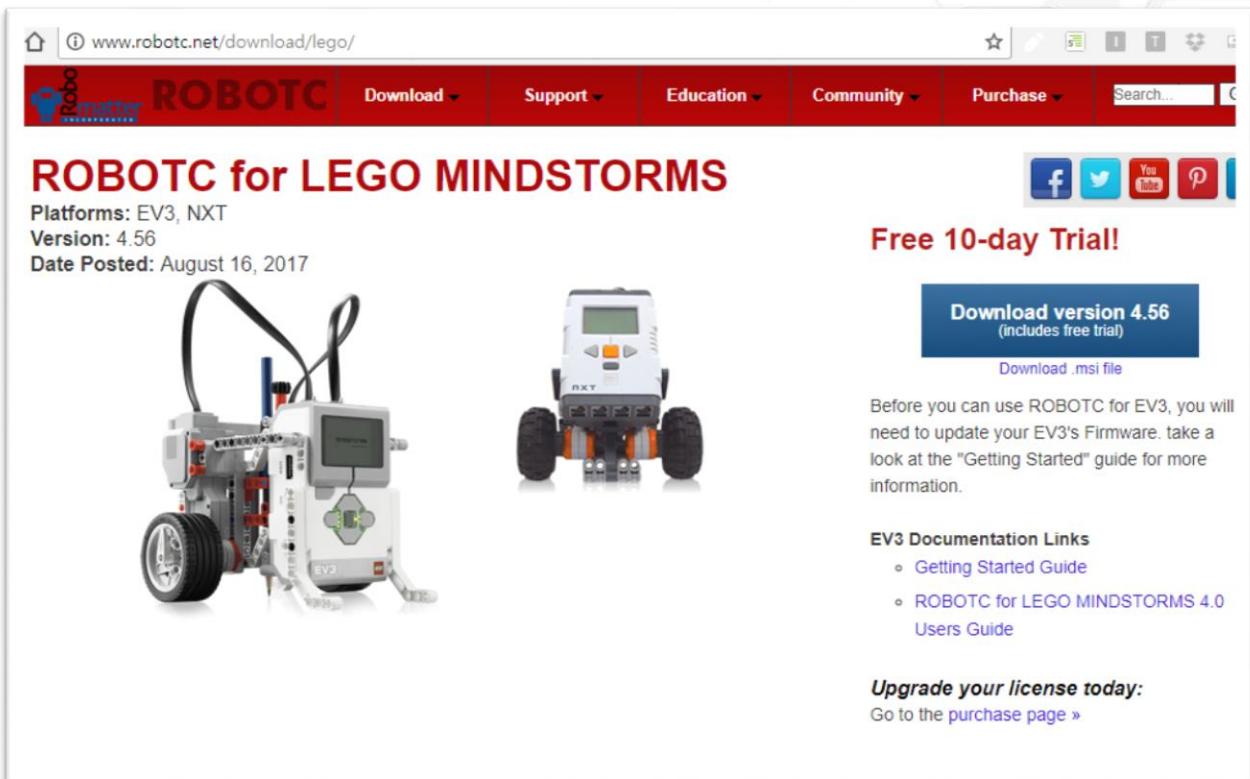
education

1부 EV3로 배우는 C언어 프로그래밍

HandsOn
Technology

1. ROBOTC 개발환경

- ROBOTC 다운로드
- ROBOTC 메뉴
- ROBOTC 시작하기



ROBOTC 다운로드

- ROBOTC 공식 홈페이지 (<http://www.ROBOTC.net>)에서 다운로드 가능
 - [Download] 메뉴의 [ROBOTC 4.X for MINDSTORMS]를 선택
 - [Download Version 4.X]를 클릭하여 설치 프로그램을 다운로드

The screenshot shows the ROBOTC website homepage. A red box highlights the 'Download' menu item, and another red box highlights the 'ROBOTC 4.x for MIND STORMS' option under it. A large red arrow points from the 'ROBOTC 4.x for MIND STORMS' link to a 'Download version 4.X' button on the right side of the page. This button is also highlighted with a red box. Below the button, there is a 'Download .msi file' link.

Free 10-day Trial!

Download version 4.X
(includes free trial)

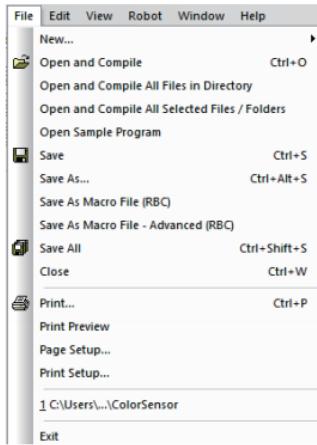
Before you can use ROBOTC for EV3, you will need to update your EV3's Firmware. Take a look at the "Getting Started" guide for more information.

EV3 Documentation Links

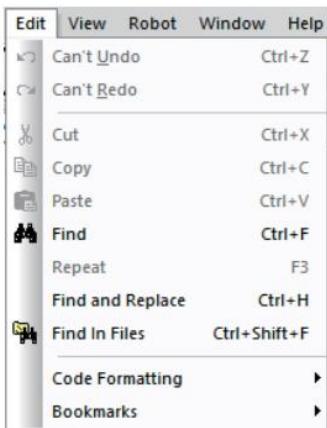
- Getting Started Guide
- ROBOTC for LEGO MINDSTORMS 4.0 Users Guide

Upgrade your license today:
Go to the purchase page »

ROBOTC 메뉴



- File 탭에서는 새 파일, 파일 열기, 예제 파일, 저장, 인쇄 등의 기능과 최근 열어본 파일 목록 등을 제공
 - 새 파일, 파일 열기, 저장은 별도의 아이콘으로 나타나 있어서 아이콘을 클릭하여 바로 실행할 수 있음



- Edit 탭에서는 실행 취소, 다시 실행 기능과 잘라내기/복사/붙여넣기, 찾기/바꾸기, 주석/들여쓰기 등의 편집 관련 기능들을 제공

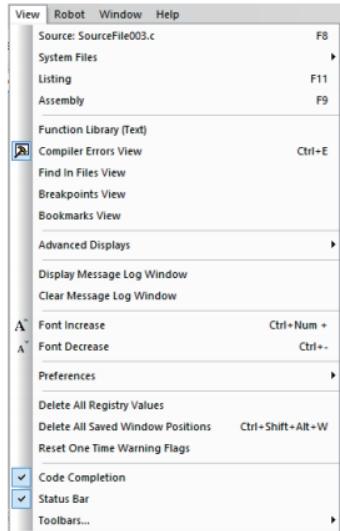


education

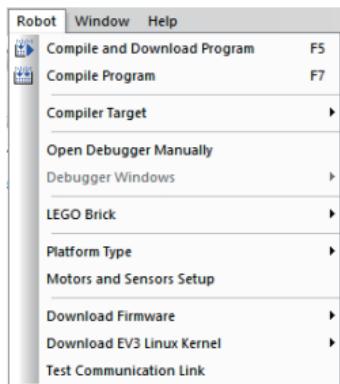
1부 EV3로 배우는 C언어 프로그래밍

HandsOn
Technology

ROBOTC 메뉴

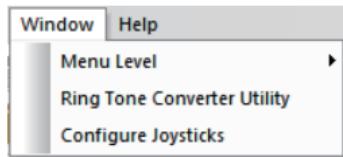


- View 탭에서는 글자 크기 설정, 상태 바 및 툴바 표시 설정 등 편집기 화면 구성과 관련된 환경을 설정할 수 있음



- Robot 탭에서는 연결할 로봇의 펌웨어 지정, 로봇과의 연결 상태 설정, 모터 및 센서 관련 설정, 컴파일 및 다운로드 기능 등을 제공

ROBOTC 메뉴



- ◆ Window 탭에서는 메뉴 레벨의 선택이 가능함
 - 'Basic', 'Expert', 'Super User' 중 하나를 선택할 수 있으며, 선택에 따라 나타나는 함수 목록의 범위가 달라짐

Tip

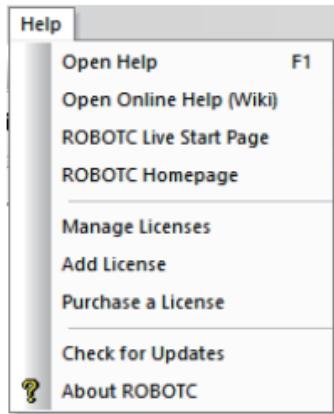
ROBOTC에서 제공하는 함수 목록

Text Functions

- ⊕ ~Control Structures
- ⊕ Battery & Power Control
- ⊕ Bluetooth Structures
- ⊕ Buttons
- ⊕ Datalog
- ⊕ Debug
- ⊕ Display
- ⊕ File Access
- ⊕ High Speed
- ⊕ IO Map Access
- ⊕ Math
- ⊕ Miscellaneous
- ⊕ Motors
- ⊕ MultiRobot
- ⊕ Semaphore
- ⊕ Sensors
- ⊕ Sensors I2C
- ⊕ Servos
- ⊕ Sound
- ⊕ Strings
- ⊕ Task Control
- ⊕ Timing
- ⊕ User Defined

- ◆ 좌측은 메뉴 레벨을 ‘Super User’로 선택 했을 때 나타나는 함수 목록이다
 - 이 함수 목록은 EV3 제어를 위해 필요한 함수 목록과 해당 함수를 사용하기 위한 형식을 알려주며, 필요한 기능에 따라 분류 되어 있음

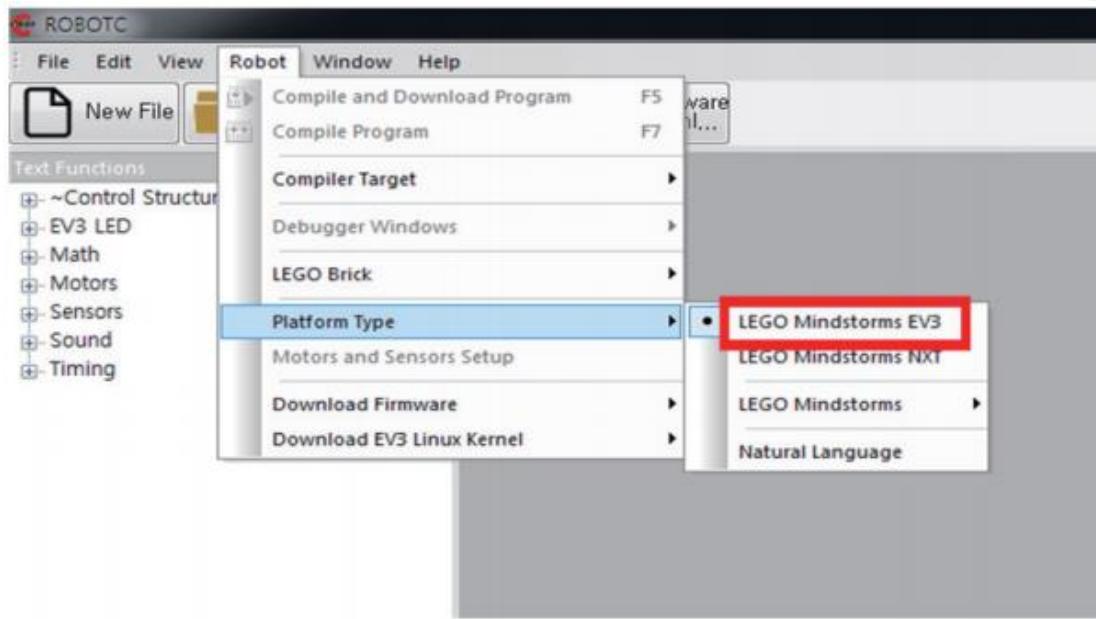
ROBOTC 메뉴



- Help 탭에서는 도움말, ROBOTC 홈페이지로의 이동, 라이선스 관리, 업데이트 확인 등의 기능을 제공
- 또한 기초 문법 및 예시 코드 등을 제공하며, 정품 소프트웨어 구매자를 위한 라이선스 활성화 및 비활성화 기능을 제공
- ROBOTC를 삭제할 경우에는 반드시 'Manage Licences'를 통해 소프트웨어의 라이선스를 비활성화('Deactivate Licences') 해야 다른 컴퓨터에서 해당 라이선스를 사용할 수 있음

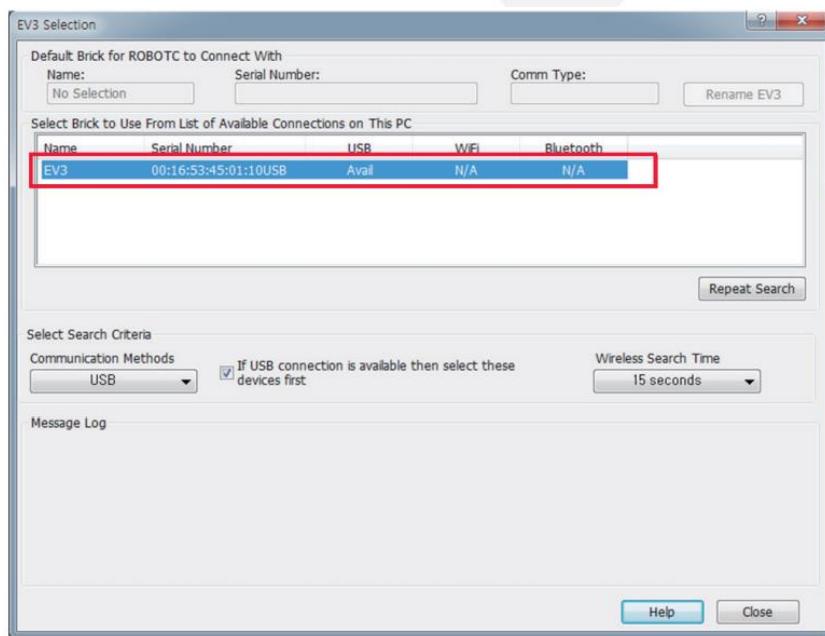
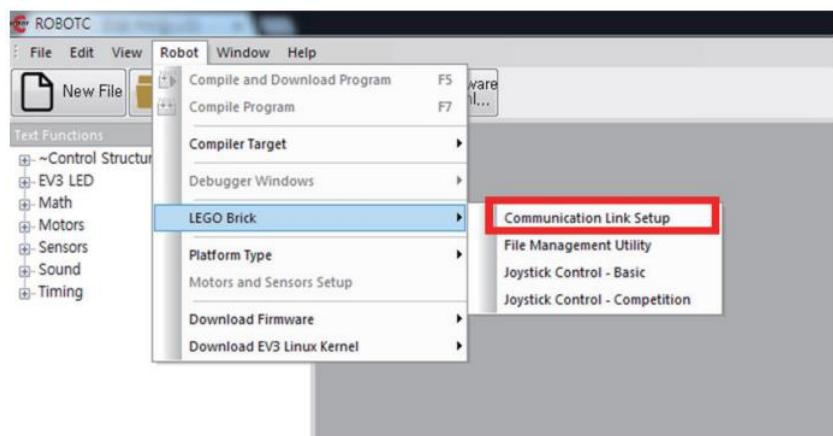
ROBOTC 시작하기

- EV3는 USB 케이블과 무선 통신을 이용하여 컴퓨터와 연결할 수 있음
- ROBOTC와 EV3를 연결하기 위해선 [Robot] - [Platform Type] 메뉴에서 'LEGO Mindstorms EV3' 플랫폼을 선택해야 함



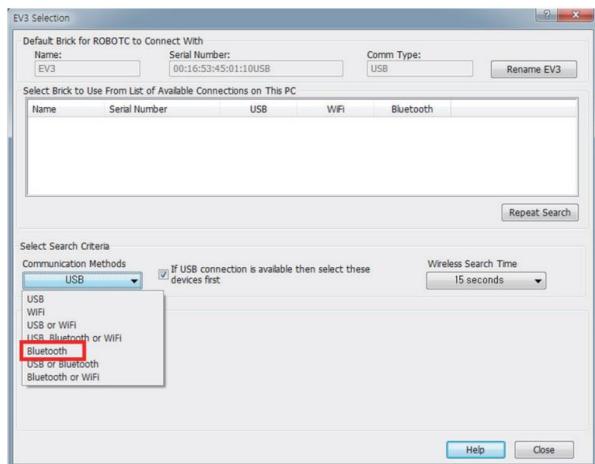
ROBOTC에 EV3 연결하기

- EV3에 연결하는 USB 케이블의 한 쪽은 Mini-USB타입으로 EV3의 포트 D 옆에 있는 Mini-USB 포트에 연결해야 함
 - 연결된 EV3는 [Robot] – [LEGO Brick] – [Communication Link Setup] 메뉴를 이용해서 확인할 수 있음



블루투스 통신 설정

- 블루투스 통신을 이용하려면 블루투스 통신이 가능한 컴퓨터가 필요
 - EV3 브릭 앱의 Bluetooth에서 ‘Visibility’, ‘Bluetooth’를 체크한 후 ROBOTC의 [Robot] - [LEGO Brick] - [Communication Link Setup] 메뉴에서 ‘Communication Methods’를 ‘Bluetooth’로 설정하고 [Repeat Search] 버튼을 클릭하면 블루투스 통신이 가능한 기기들을 찾을 수 있음

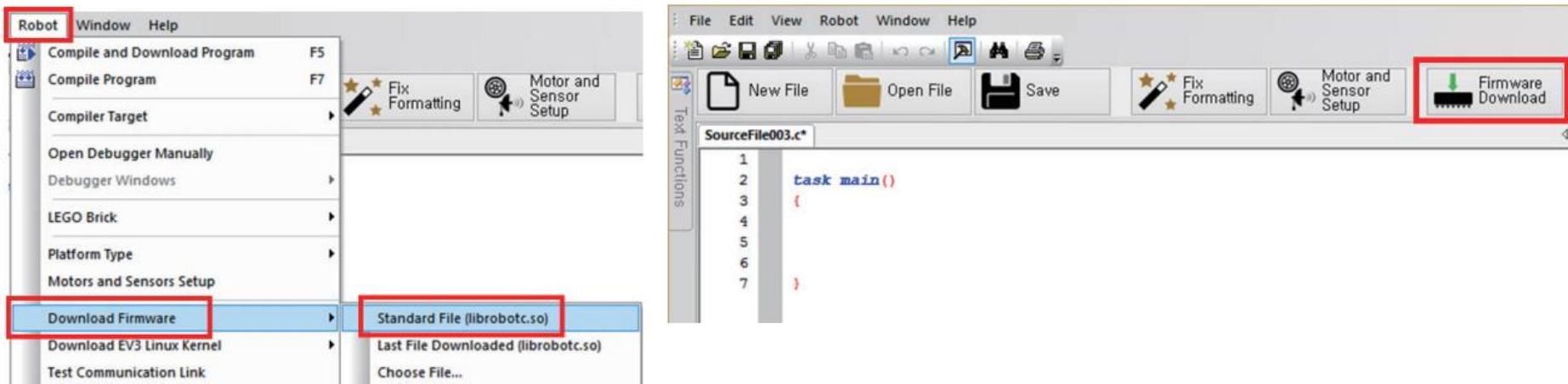


- 자신의 EV3 기기를 식별하기 위해서는 자신만의 EV3 이름을 지어주는 것이 좋음
- EV3의 이름을 변경할 때에는 EV3가 연결된 상태에서 [Rename EV3] 버튼을 클릭하면 됨

EV3 펌웨어 다운로드

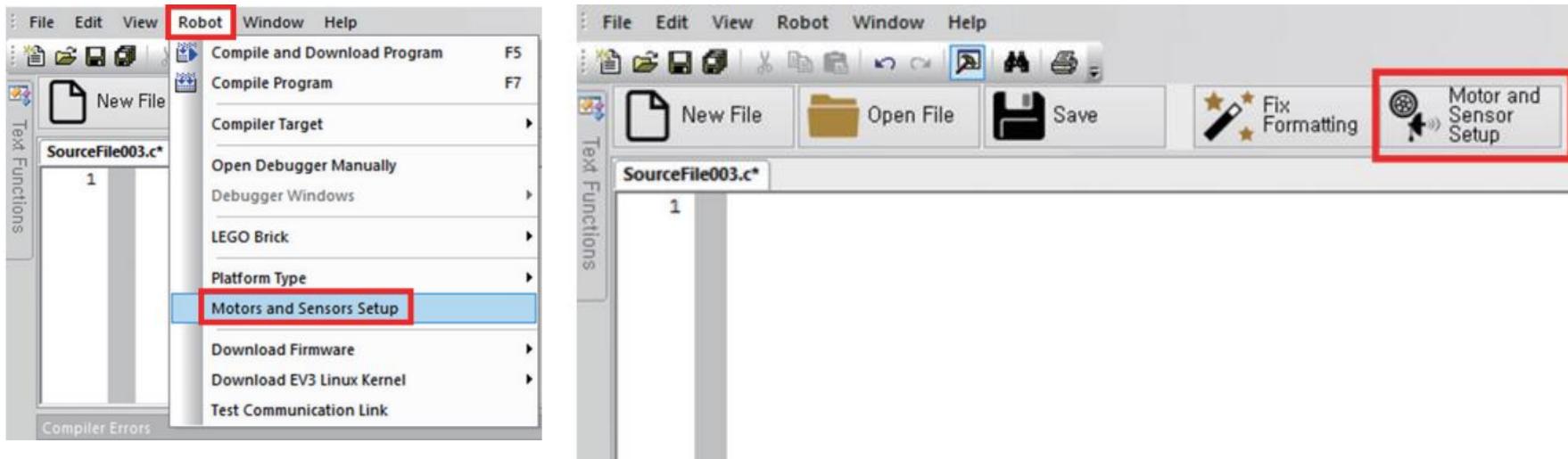
Platform Type 설정 --> Linux Kernel를 설치 --> Download Firmware

- EV3 펌웨어는 다른 기기와의 인터페이스 및 사용자와의 인터페이스를 제공
 - EV3에 설치된 펌웨어와 ROBOTC의 펌웨어 버전이 일치해야 다운로드 된 프로그램이 EV3에 다운로드 되어 실행될 때 제대로 동작할 수 있음
 - 펌웨어는 [Robot] - [Download Firmware] 메뉴를 선택하거나, 아이콘 중 [Firmware Download] 버튼을 클릭하여 다운로드 할 수 있음



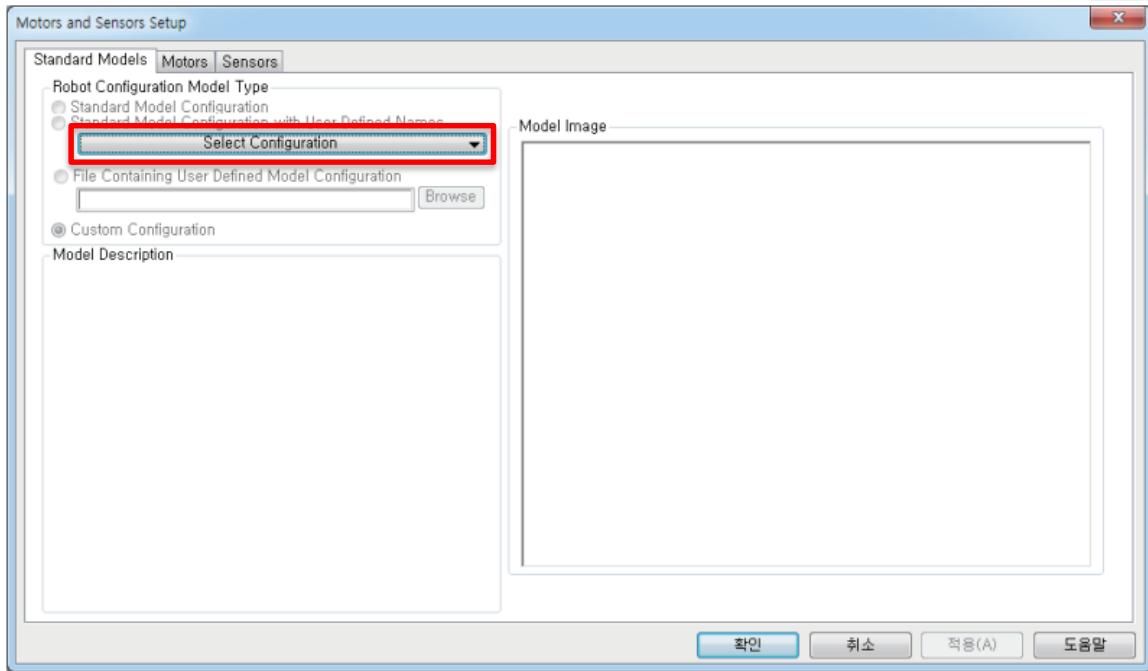
모터 및 센서 설정

- [Robot] – [Motors and Sensors Setup] 메뉴를 선택하거나 [Motors and Sensors Setup] 아이콘을 클릭하여 모터와 센서의 연결 포트와 종류를 설정
 - 이 메뉴는 ROBOTC와 로봇이 연결되어 있고, 소스 코드 작성을 위해 새 파일을 생성하거나 기존 파일을 불러와야 활성화됨



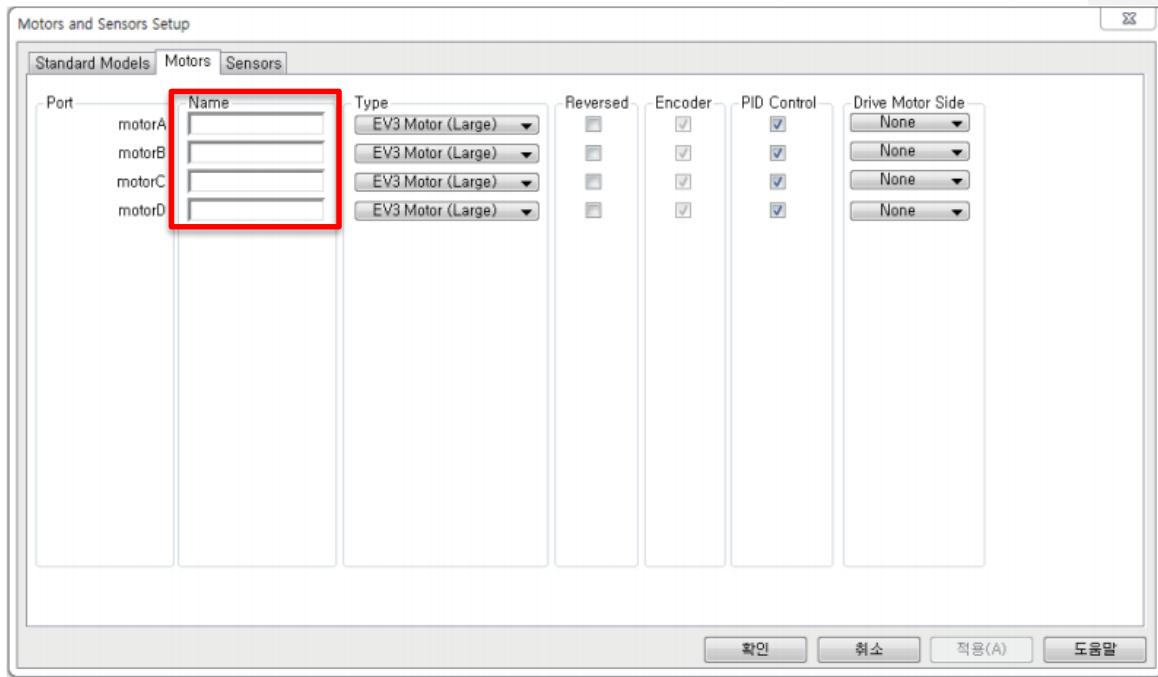
모터와 센서 자동 설정

- LEGO가 공식 지원하는 로봇 모델의 모터와 센서가 미리 설정되어 있음
 - ‘Standard Models’ 탭에 나타나는 항목들 중 ‘Select Configuration’ 콤보 박스에서 원하는 로봇 모델을 선택하면 자동으로 그에 알맞은 로봇의 모터와 센서가 설정됨



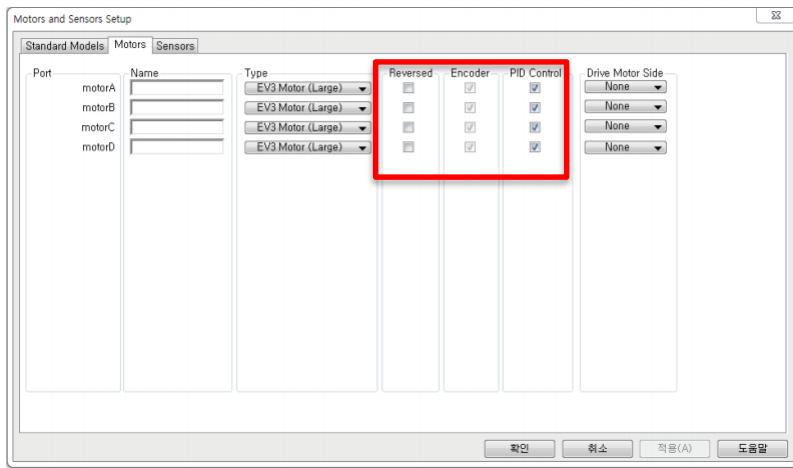
모터 이름 설정

- 'Motors' 탭에서는 프로그램 상에서 사용할 모터의 이름, 종류 등을 설정할 수 있음
 - 왼쪽 바퀴가 연결된 모터는 'lm(left motor)'으로, 오른쪽 바퀴가 연결된 모터는 'rm(right motor)'으로 설정하는 등의 활용으로 코드의 가독성을 높일 수 있음



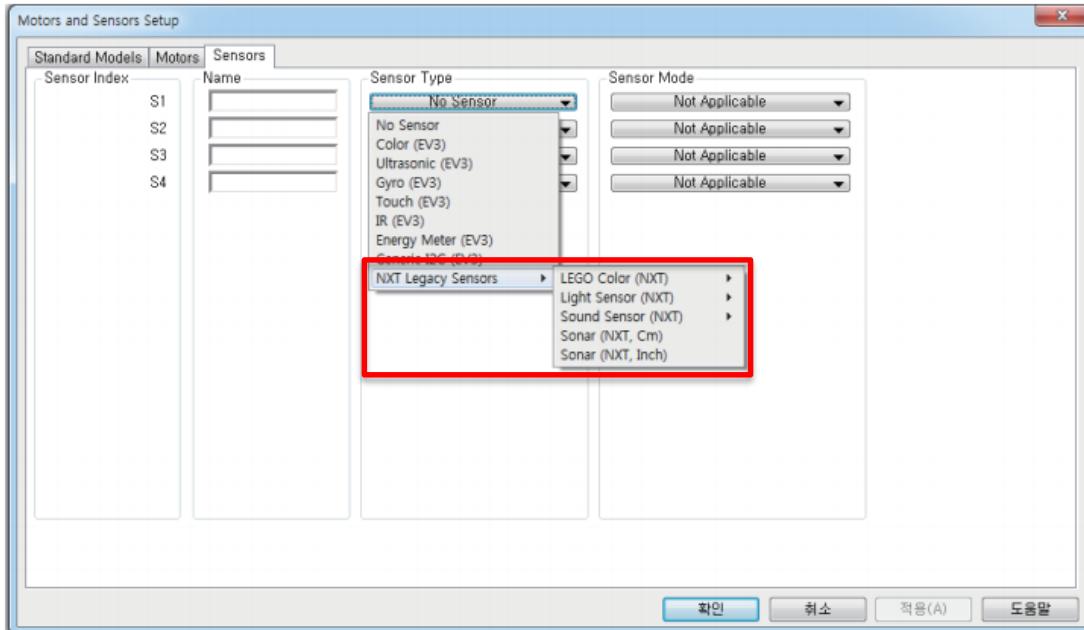
모터 추가 설정

- 모터의 종류는 NXT Motor와 **EV3 Motor(Large)**, EV3 Motor(Medium) 3가지 중 하나를 선택할 수 있음
 - 'Reversed'를 체크하면 모터의 회전 방향이 반대가 됨
 - 'PID Control'을 체크하면 모터가 외부 변인에 영향 받지 않고 설정한 모터 값을 유지할 수 있도록 소프트웨어적으로 모터 컨트롤을 하게 됨



센서 설정

- ‘Sensors’ 탭에서는 센서 이름, 센서 종류 등을 설정 할 수 있음
 - Sensor Type은 센서의 종류를 설정
 - EV3 센서 이외에도 NXT용 센서를 사용하기 위한 설정도 가능함



Driving base 모터/센서 설정 완료

db_start.c

```

1 #pragma config(Sensor, S1,      ts,          sensorEV3_Touch)
2 #pragma config(Sensor, S2,      gs,          sensorEV3_Gyro)
3 #pragma config(Sensor, S3,      cs,          sensorEV3_Color)
4 #pragma config(Sensor, S4,      us,          sensorEV3_Ultrasonic)
5 #pragma config(Motor,  motorB,    lm,          tmotorEV3_Large, PIDControl, driveLeft, encoder)
6 #pragma config(Motor,  motorC,    rm,          tmotorEV3_Large, PIDControl, driveRight, encoder)
7 //!!!Code automatically generated by 'ROBOTC' configuration wizard      !!!//
8
9 task main()
10 {
11
12
13
14 }

```



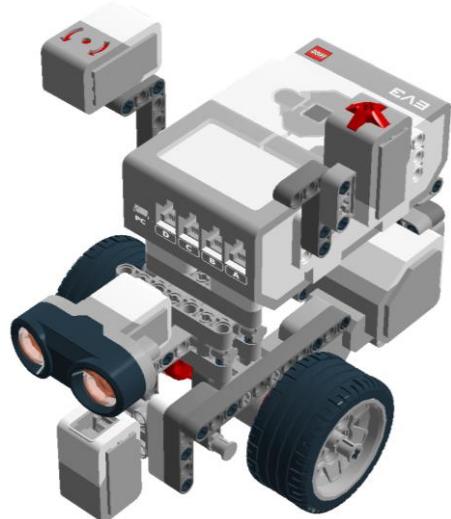
education

1부 EV3로 배우는 C언어 프로그래밍

HandsOn
Technology

2. ROBOTC 기초

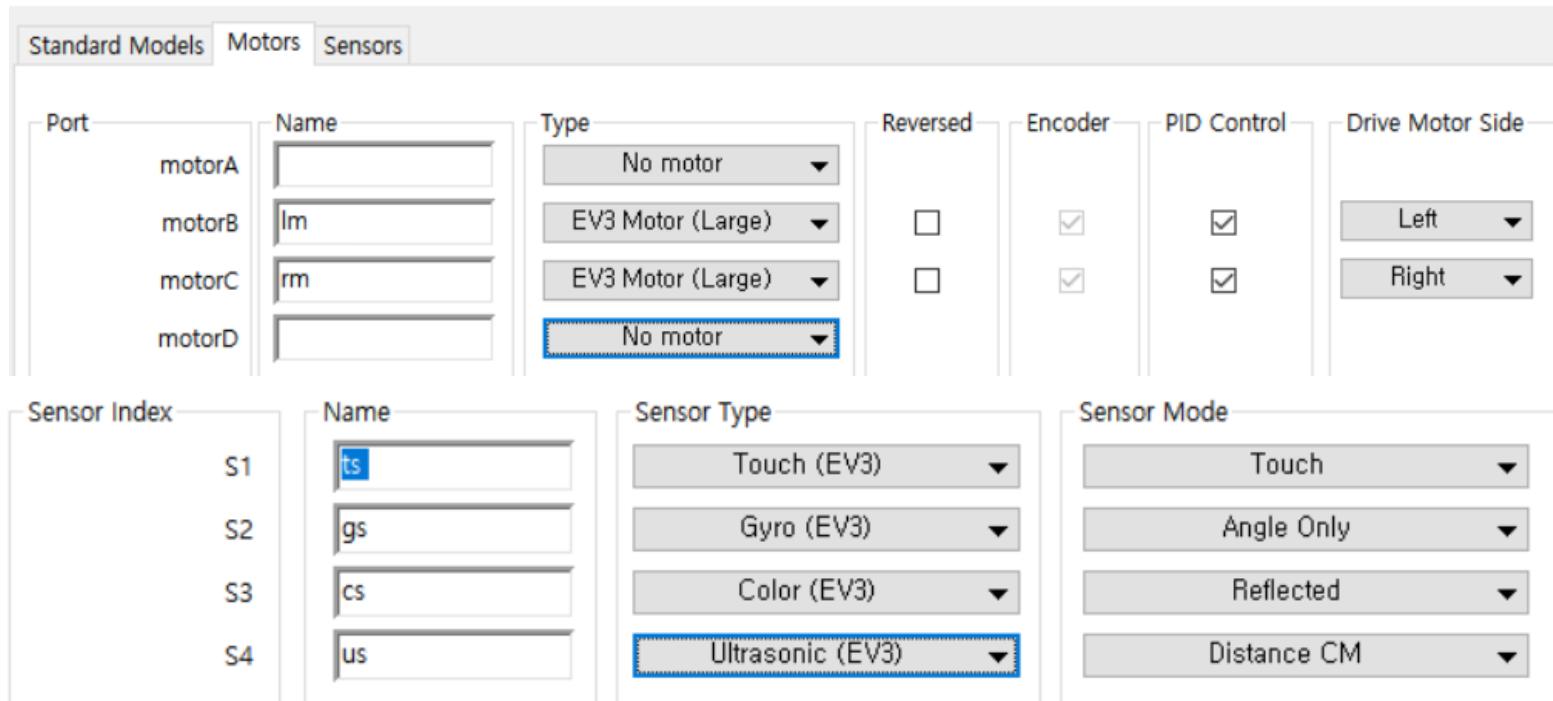
- ROBOTC 규칙
- 시간(지연) 함수
- 데이터 저장 및 계산
- 조건 선택/실행
- 반복 실행
- 함수
- 배열



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



ROBOTC 규칙

- 최소 하나의 **main()** 함수가 필요함
 - task main() 함수 안에 실행하고자 하는 프로그램을 기록
 - main() 함수의 바깥에 추가적으로 함수를 설정하여 이용하기도 함
- 전처리 구문은 함수 밖에 따로 선언
 - 전처리 구문 : task main() 함수보다 먼저 처리되는 구문
 - **#include, #define, #pragma** 등과 같이 '#'으로 시작하고 ';'을 사용하지 않음
 - **#define**은 기호를 정의하여 프로그램 전체에 대해 대체하도록 하는 기능을 함
 - 매크로라고도 하며, 매크로가 실행되면 정의된 기호가 상수 값을 대체

로봇의 전진(매크로 상수)

줄번호	로봇의 전진(매크로 상수)
01	#pragma config(Motor, motorB, lM, tmotorEV3_Large, PIDControl, encoder)
02	#pragma config(Motor, motorC, rM, tmotorEV3_Large, PIDControl, encoder)
03	
04	<u>#define MOTOR 50</u>
05	
06	task main()
07	{
08	setMotorSpeed(rM, MOTOR);
09	setMotorSpeed(lM, MOTOR);
10	sleep(3000);
11	}
줄번호	코드 설명
04	'MOTOR'를 50으로 정의한다.
08~10	3초 동안 앞으로 전진한다.

ROBOTC 규칙

- ROBOTC의 컴파일러는 대·소문자를 엄격하게 구분함
 - 특정한 라이브러리 함수를 이용하기 위해서 대·소문자를 구분해야함
ex) nMotorEncoder (O) nmotorencoder (X)
- ROBOTC 프로그램의 문장의 끝에는 세미콜론(;)이 들어가야 함
 - 세미콜론(;)이 없으면 컴파일러가 문장의 끝을 인식하지 못해 오류가 생길 수 있음
- 주로 사용되는 단어는 색이 다르게 나타남
 - ROBOTC가 인식하는 자주 사용되는 키워드는 색상이 결정되어 표시됨
ex) 'int', 'task main' → 파랑색 'motor' → 보라색

ROBOTC 규칙

- 코드는 기본적으로 1번 줄부터 실행됨
 - 프로그램은 1번 줄부터 순차적으로 실행되며 함수가 호출될 경우 호출된 함수의 1번 줄부터 실행됨
 - 함수가 종료되면 main() 함수 내의 함수가 호출되었던 부분으로 돌아감
 - 'Step Into' 버튼을 통해 코드가 진행되는 과정을 확인할 수 있음
- 프로그램의 가독성을 높여야 함
 - 프로그램을 작성하고, 수정, 보안하기 위해서는 프로그램의 가독성이 중요함
 - 프로그램의 가독성을 높이기 위해서는 적절한 공백과 줄바꿈 필요
 - 코드 설명을 추가하기 위해 코드 옆에 ‘//’를 이용하여 한 줄 주석을 달 수 있음

시간(지연) 함수

- 시간 함수 : 특정 시간동안 다음 코드로 넘어가지 않고 지연시키는 함수
 - 모터값을 유지시키거나 코드의 주기를 조절할 때 사용, sleep() 함수가 대표적
 - 괄호 안에는 머물고자 하는 시간을 입력하며 시간 단위는 0.001초(1ms)
 - sleep() 함수와 동일한 역할을 하는 함수에는 wait1Msec() 함수가 있음

```
sleep(1000); // 1msec * 1000 = 1sec 동안 머문다.
```

로봇의 전진(시간 함수 없음)

줄번호	로봇의 전진(시간 함수 미사용)
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(rm, 70);
07	setMotorSpeed(lm, 70);
08	}
줄번호	코드 설명
06	모터 rm의 모터 값을 70으로 지정한다.
07	모터 lm의 모터 값을 70으로 지정한다. → 로봇이 전진하지 않음

로봇의 전진(시간 함수 사용)

줄번호	로봇의 전진(시간 함수 사용)
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(rM, 70);
07	setMotorSpeed(lM, 70);
08	<u>sleep(5000);</u>
09	}
줄번호	코드 설명
06	모터 rM의 모터 값을 70으로 지정한다.
07	모터 lM의 모터 값을 70으로 지정한다.
08	5초 동안 유지한다. → 로봇이 전진함

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

데이터 저장 및 계산

- 상수 : 변수와 달리 한번 값이 정해지면 변하지 않음

<#define을 이용한 상수 선언>

```
#define PI 3.14 // PI라는 상수를 3.14로 선언한다.
```

```
#define HI "Hello, world" // HI라는 상수를 "Hello, world"라는 문자열로 선언한다.
```

<const를 이용한 상수 선언>

```
const double PI = 3.14;
```

```
const char HI[13] = "Hello, world";
```

데이터 저장 및 계산

- 자료형 : 데이터의 종류와 저장(표현) 범위
 - 데이터의 종류에는 정수, 실수, 문자 등이 있고, 표현하고자 하는 값의 범위도 다양함
 - 데이터의 자료형을 지정해 저장 공간의 크기, 값의 범위, 데이터의 종류 등을 결정 가능

자료형		의미	바이트 수
정수형	int	정수	4
	unsigned int	0을 포함한 양수인 정수	4
실수형	float	실수	4
문자형	char	문자	1

정수형 변수

- 정수 형태의 데이터를 저장하는 공간
 - ROBOTC에서 가장 많이 사용하는 변수의 종류
- 〈정수형 변수의 선언〉

```
int x;  
unsigned int x;
```

〈정수형 변수의 초기화〉

```
int x = 12;  
  
int y;  
y = 12;
```

정수형 변수

- 정수 형태의 데이터를 저장하는 공간
 - ROBOTC에서 가장 많이 사용하는 변수의 종류
- 〈정수형 변수의 출력〉

```
displayBigTextLine(1, "x is %d", x); // int형 변수 출력  
displayBigTextLine(1, "x is %u", x); // unsinged int형 변수 출력
```

정수형 변수의 선언과 출력

줄번호	정수형 변수값의 출력
01	task main()
02	{
03	<u>int x = 12;</u>
04	x = 5;
05	displayBigTextLine(1, "x is <u>%d</u> ", x);
06	sleep(5000);
07	}
줄번호	코드 설명
03	정수형 변수 x를 선언하고 변수를 12로 초기화한다.
04	변수 x의 값을 5로 변경한다.
05	EV3 디스플레이 1번 라인에 "x is 5"를 출력한다.
06	5초 동안 유지시킨다.

실수형 변수

- 실수 형태의 데이터를 저장하는 공간

〈실수형 변수의 선언과 초기화〉

```
float x = 3.14;  
float y;  
y = 9.1;
```

〈실수형 변수의 출력〉

```
displayBigTextLine(1, "x is %f", x); // float형 변수 출력
```

실수형 변수

- 실수 형태의 데이터를 저장하는 공간

〈실수형 변수의 소수점 출력〉

```
displayBigTextLine(1, "x is %.2f", x); // float형 변수를 소수점 2번째 자리까지 출력
```

실수형 변수의 선언과 출력

줄번호	실수형 변수값의 출력
01	task main()
02	{
03	<u>float x;</u>
04	x = 3.141592;
05	displayBigTextLine(1, "x is <u>%.2f</u> ", x);
06	sleep(5000);
07	}
줄번호	코드 설명
03	실수형 변수 x를 선언한다.
04	변수 x를 3.141592로 초기화한다.
05	변수 x를 소수점 2번째 자리까지(3번째 자리에서 반올림) 출력한다.
06	EV3 디스플레이 1번 라인에 "x is 3.14"를 출력한다.
06	5초 동안 유지시킨다.

문자형 변수

- 1개의 문자를 저장하기 위한 공간
 - 컴퓨터는 모든 것을 숫자로 받아들임
 - 'a'라는 문자를 입력했을 때, 컴퓨터에는 97이라는 숫자로 저장됨
 - 따라서 문자형 변수에 문자 'a'를 저장한 결과와 숫자 97을 저장한 결과가 같음

문자형 변수

- ◆ 아스키 코드(ASCII) : 문자를 컴퓨터 내부에 저장하기 위해 숫자에 대응시킨 결과를 제시한 표

Dec	Hex	문자												
40	28	(60	3C	<	80	50	P	100	64	d	120	78	x
41	29)	61	3D	=	81	51	Q	101	65	e	121	79	y
42	2A	*	62	3E	>	82	52	R	102	66	f	122	7A	z
43	2B	+	63	3F	?	83	53	S	103	67	g	123	7B	{
44	2C	.	64	40	@	84	54	T	104	68	h	124	7C	
45	2D	-	65	41	A	85	55	U	105	69	i	125	7D	}
46	2E	.	66	42	B	86	56	V	106	6A	j	126	7E	~
47	2F	/	67	43	C	87	57	W	107	6B	k	127	7F	DEL
48	30	0	68	44	D	88	58	X	108	6C	l			
49	31	1	69	45	E	89	59	Y	109	6D	m			
50	32	2	70	46	F	90	5A	Z	110	6E	n			
51	33	3	71	47	G	91	5B	[111	6F	o			
52	34	4	72	48	H	92	5C	\	112	70	p			
53	35	5	73	49	I	93	5D]	113	71	q			
54	36	6	74	4A	J	94	5E	^	114	72	r			
55	37	7	75	4B	K	95	5F	_	115	73	s			
56	38	8	76	4C	L	96	60	'	116	74	t			
57	39	9	77	4D	M	97	61	a	117	75	u			
58	3A	:	78	4E	N	98	62	b	118	76	v			
59	3B	;	79	4F	O	99	63	c	119	77	w			

문자형 변수

- 1개의 문자를 저장하기 위한 공간

〈문자형 변수의 선언〉

```
char x;
```

〈문자형 변수의 초기화〉

- 문자형 변수에 변수값을 저장할 때에는 저장할 문자의 아스키 코드값을 저장해야 함
- 문자에 작은 따옴표를 써웠을 경우 그 문자의 아스키 코드값을 나타냄
- 특정 문자의 아스키 코드값을 직접 저장 가능

```
char x = 'a'; // 'a' 대신 97을 넣어도 된다.
```

```
char y;
```

```
y = 'a';
```

문자형 변수

- 1개의 문자를 저장하기 위한 공간

〈문자형 변수의 출력〉

```
displayBigTextLine(1, "x is %c", x);
```

문자형 변수의 선언과 출력

줄번호	문자형 변수값의 출력
01	task main()
02	{
03	<u>char x = 'a';</u>
04	x = 98;
05	displayBigTextLine(1, "x is <u>%c</u> ", x);
06	sleep(5000);
07	}
줄번호	코드 설명
03	문자형 변수 x를 선언하고 변수를 'a'로 초기화한다.
04	변수 x의 값을 아스키코드 98에 해당하는 'b'로 변경한다.
05	EV3 디스플레이 1번 라인에 "x is b"를 출력한다.
06	5초 동안 유지시킨다.

지역 변수 (local variable)

- 함수 내부에서 선언된 변수
 - 변수가 선언될 때 생성되고, 함수가 종료되면 동시에 소멸됨

줄번호	지역 변수값의 출력
01	task main()
02	{
03	<u>int x = 3;</u>
04	displayBigTextLine(1, "x is %d", x);
05	sleep(5000);
06	}
줄번호	코드 설명
03	정수형 변수 x를 선언한다. main() 함수 안에서 선언하였으므로 지역 변수이다.
04	1번 라인에 "x is 3"을 출력한다.
05	5초간 유지한다.

전역 변수 (global variable)

- 함수 외부에서 선언된 변수
 - 프로그램 전체에서 사용할 수 있어, 여러 함수에서 공유될 수 있음

2-8	전역 변수값의 출력
01	<u>int x = 12;</u>
02	
03	task main()
04	{
05	displayBigTextLine(1, "x is %d", x);
06	sleep(5000);
07	}
줄번호	코드 설명
01	정수형 변수 x를 선언한다. main() 함수 밖에서 선언하였으므로 전역 변수이다.
05	첫 번째 줄에 "x is 12"를 출력한다.
06	5초 동안 유지한다.

형 변환 (casting)

- 변수값의 자료형을 바꾸는 것
 - 다른 자료형 변수와 연산을 할 때 주로 사용함
 - 변환하고자 하는 데이터 앞에 바꾸고자 하는 자료형을 써주면 형 변환이 이루어짐

```
x = (int)3.79; // 실수 3.79를 int(정수)형으로 변환하여 3을 x에 저장  
x = (double)y; // 변수 y의 값을 double(실수)형으로 변환하여 x에 저장
```

변수 값의 (자료)형 변환

2-9	형 변환 결과의 출력
01	task main()
02	{
03	float x = 3.79;
04	int y;
05	y = <u>(int)x</u> ;
06	displayBigTextLine(1, "y is %d", y);
07	displayBigTextLine(3, "x is %.2f", x);
08	sleep(5000);
09	}
줄번호	코드 설명
03	실수형 변수 x를 선언하고 변수를 3.79로 초기화한다.
04	정수형 변수 y를 선언한다.
05	변수 x의 값을 정수형으로 형 변환하여 변수 y에 저장한다.
06	EV3 디스플레이 1번 라인에 "y is 3"을 출력한다.
07	EV3 디스플레이 3번 라인에 "x is 3.79"를 출력한다. x 자체의 변수값은 바뀌지 않은 것을 알 수 있다.
08	5초간 유지시킨다.

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

연산자

- **증감 연산자** : 변수의 값을 1만큼 증가시키거나 감소시키는 연산자
 - 어떤 상황이 몇 번 생기는지 셀 때 자주 사용됨

연산자	연산식	의미
증가	$x++$	x 를 1 증가시킨다
감소	$x--$	x 를 1 감소시킨다

연산자

- **대입 연산자** : 변수에 데이터를 저장하기 위해 사용되는 연산자
 - 산술 연산자와 결합하여 복합 대입 연산자로 표현 가능

연산자	연산식	의미
대입 연산자	$x = 3$	3을 x에 저장한다
복합 대입 연산자	$x += 3$	$x = x + 3$
	$x -= 3$	$x = x - 3$
	$x *= 3$	$x = x * 3$
	$x /= 3$	$x = x / 3$

연산자

- **비교 연산자** : 두 변수나 상수를 비교하는 연산자

연산식	의미
$x == y$	x 와 y 가 서로 같은가
$x != y$	x 와 y 가 서로 다른가
$x < y$	x 가 y 보다 작은가
$x > y$	x 가 y 보다 큰가
$x <= y$	x 가 y 보다 작거나 같은가
$x >= y$	x 가 y 보다 크거나 같은가

연산자: && , || , !

- 논리 연산자 : 참과 거짓을 판단하는 연산자
 - 참일 경우에는 1, 거짓일 경우에는 0을 나타냄

연산자	연산식	의미
AND 연산자	$x \&\& y$	x 와 y 모두 참일 경우 참 x 나 y 중 하나만 거짓이어도 거짓
OR 연산자	$x y$	x 나 y 중 하나만 참이어도 참 둘 다 거짓일 경우 거짓
NOT 연산자	$!x$	x 가 거짓일 경우, 참 x 가 참일 경우, 거짓

논리 연산자를 이용한 전진과 후진

2-10

논리 연산자를 이용한 전진과 후진

```

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     int x = 3;
07     int y = 6;
08
09     if(x<5 && y<5)
10    {
11        playSound(soundBeepBeep);
12        sleep(1000);
13    }
14
15    if(x<5 || y<5)
16    {
17        setLEDColor(7);
18        sleep(1000);
19    }
20
21    if( !(x+y < 5) )
22    {
23        setMotorSpeed(rm, 30);
24        setMotorSpeed(lm, 30);
25        sleep(1000);
26    }
27 }
```

09~13 | x가 5보다 작고, y도 5보다 작으면 1초 동안 EV3가 소리를 낸다.

15~19 | x가 5보다 작거나 y가 5보다 작으면 1초 동안 EV3 LED에 초록색이 반짝인다.

21~26 | x+y가 5보다 작은 것이 아니라면 로봇이 1초 동안 전진한다.

Help

MadCap Help Viewer V6.3

File Edit View Tools Window Help

TOC

- Welcome
- > ROBOTC Installation and Activation
- > Getting Started – Physical Robots
- > Getting Started – Virtual Worlds
- > ROBOTC Language Progression
- > ROBOTC Interface
- > ROBOTC Debugger
- > General C Programming
- > Command Library – LEGO NXT
- > Command Library – LEGO EV3
 - > Graphical
 - > Natural Language
- > ROBOTC
 - > Battery and Power Control
 - > Buttons
 - > Datalogging
 - > EV3 LED
 - setLEDColor**
 - > Joystick Control
 - > LCD Display
 - > Motors
 - > Sensors
 - > Sounds
 - > Task Control
 - > Timing

setLEDColor

Help Topic

- There are 10 options for TEV3LEDpatterns:

LED_BLACK:	Turns the LED off.
LED_GREEN:	Turns the LED on to solid green.
LED_RED:	Turns the LED on to solid red.
LED_ORANGE:	Turns the LED on to solid orange.
LED_GREEN_FLASH:	Turns the LED on to blinking green.
LED_RED_FLASH:	Turns the LED on to blinking red.
LED_ORANGE_FLASH:	Turns the LED on to blinking orange.
LED_GREEN_PULSE:	Turns the LED on to pulsing green.
LED_RED_PULSE:	Turns the LED on to pulsing red.
LED_ORANGE_PULSE:	Turns the LED on to pulsing orange.

Code Sample

```
//Infinite loop
while(true)
{
    //Flash the LED Green for two seconds
    setLEDColor(ledGreenFlash);
    sleep(2000);

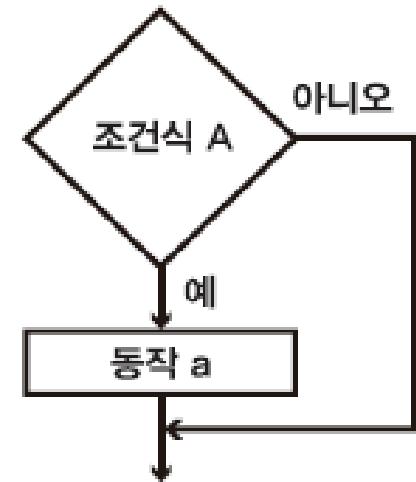
    //Pulse the LED Red for two seconds
    setLEDColor(ledRedPulse);
    sleep(2000);

    //Turns the LED solid Orange for two seconds
    setLEDColor(ledOrange);
    sleep(2000);
}
```

if문

- 만약 ‘조건식 A’를 만족한다면, ‘동작 a’를 실행함

```
if(조건식 A)
{
    동작 a;
}
```



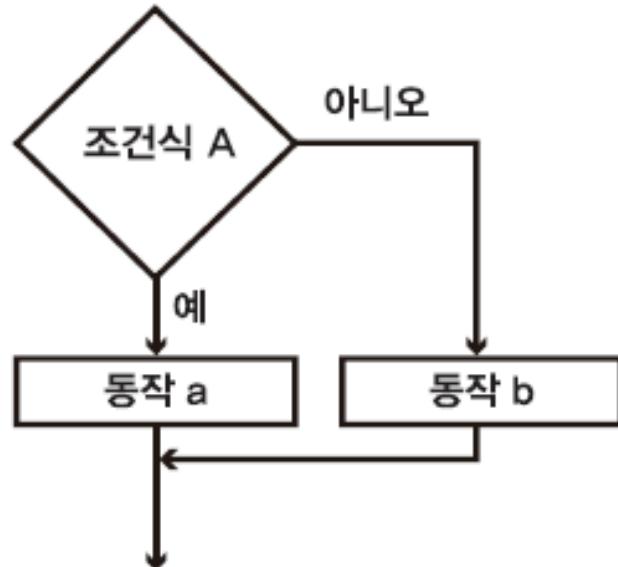
터치 센서를 이용한 후진

2-11	터치 센서를 이용한 후진(if문)
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	<u>if(getTouchValue(ts) == 1)</u>
08	{
09	setMotorSpeed(lm, -50);
10	setMotorSpeed(rm, -50);
11	sleep(2000);
12	}
13	}
줄번호	코드 설명
07	터치 센서가 눌려있을 때 하위의 명령을 수행한다.
09~11	2초간 50의 모터 값으로 후진한다.

if~else문

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

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



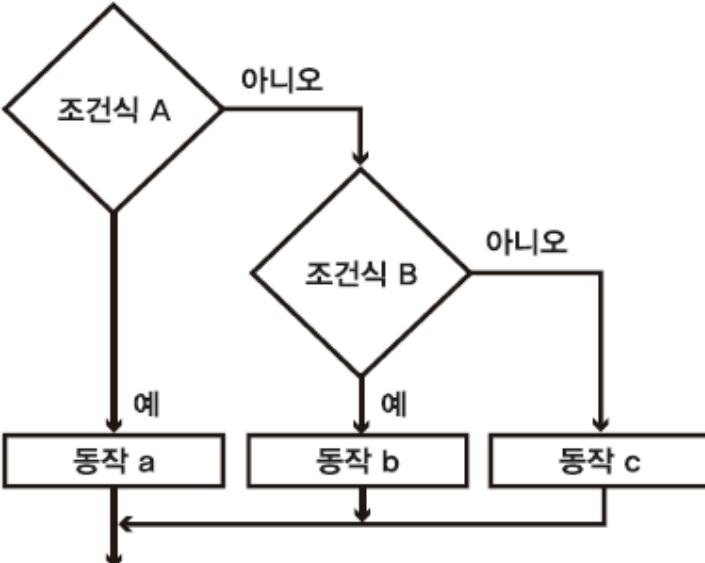
터치 센서를 이용한 전진과 후진

줄번호	터치 센서를 이용한 주행(if~else문)
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	if(getTouchValue(ts) == 1)
08	{
09	setMotorSpeed(lm, -50);
10	setMotorSpeed(rm, -50);
11	sleep(2000);
12	}
13	else
14	{
15	setMotorSpeed(lm, 50);
16	setMotorSpeed(rm, 50);
17	sleep(2000);
18	}
19	}
줄번호	코드 설명
07	터치 센서가 눌려있을 때 하위의 명령을 수행한다.
09~11	2초간 50의 모터 값으로 후진한다.
13	7번 줄의 if문이 거짓일 경우 하위의 명령을 수행한다.
15~17	2초간 50의 모터 값으로 전진한다.

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

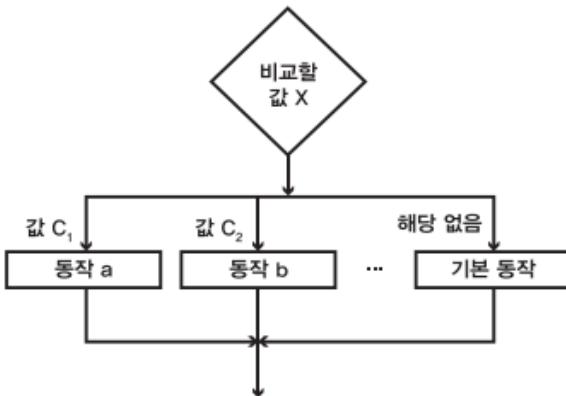
9번 줄과 15번 줄의 if문이 모두 거짓일 경우 50의 모터 값으로 전진한다.

switch~case문

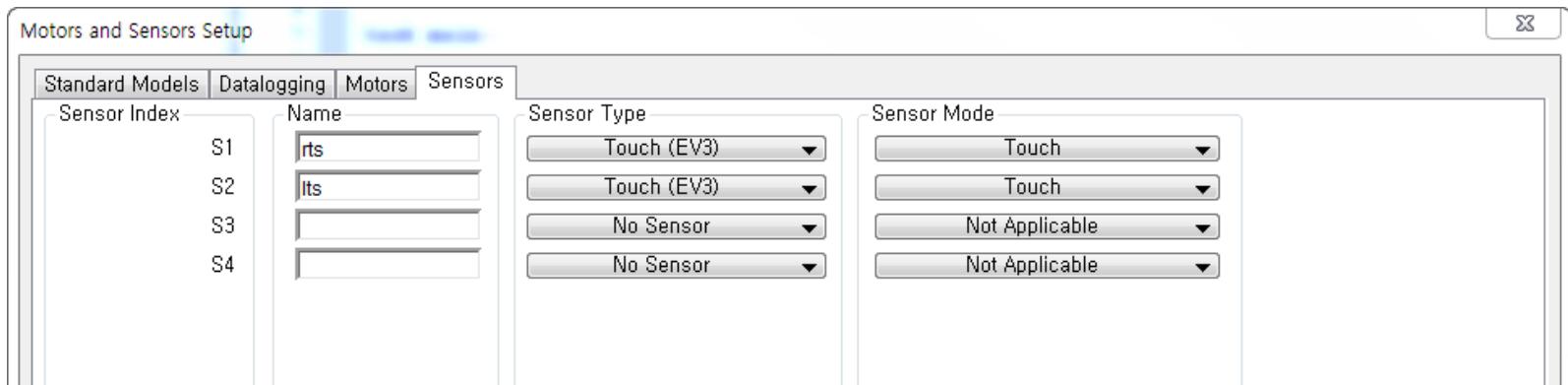
- 비교할 값 X가 C_1 이면 ‘동작 a’를, C_2 이면 ‘동작 b’를 실행하고,
(중략), 모두 아니면 ‘기본 동작’을 실행함 아니면 기본 동작 실행

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

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

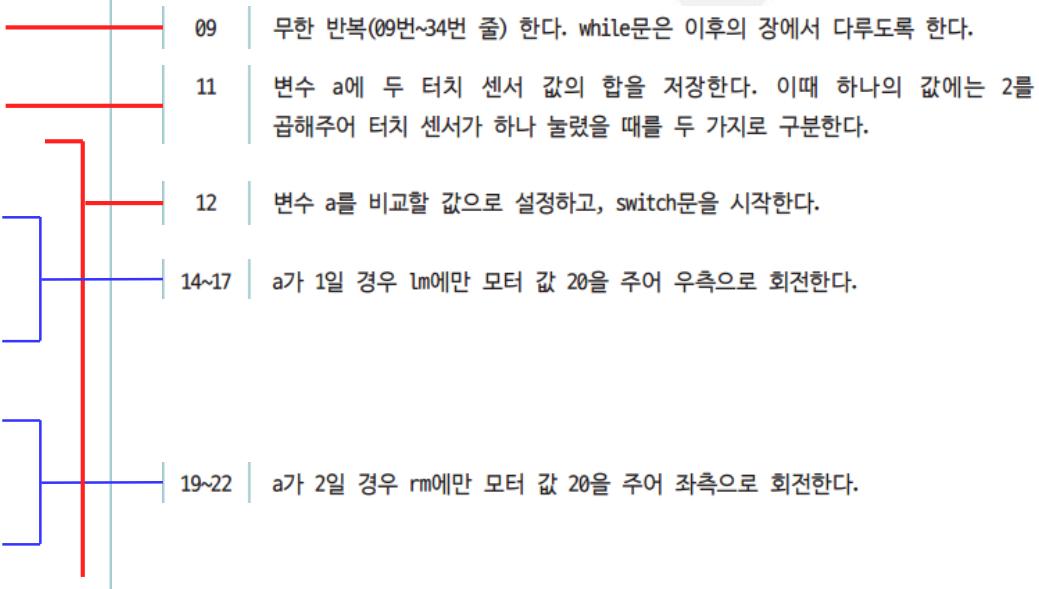


모터 및 센서 설정



터치 센서를 이용한 주행

2-15	터치 센서를 이용한 주행(switch~case문)
01	#pragma config(Sensor, S1, rts, sensorEV3_Touch)
02	#pragma config(Sensor, S2, lts, sensorEV3_Touch)
03	#pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
04	#pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
05	
06	task main()
07	{
08	int a;
09	while(1)
10	{
11	a = getTouchValue(rts) + getTouchValue(lts) * 2;
12	switch(a)
13	{
14	case 1:
15	setMotorSpeed(lm, 20);
16	setMotorSpeed(rm, 0);
17	break;
18	
19	case 2:
20	setMotorSpeed(lm, 0);
21	setMotorSpeed(rm, 20);
22	break;
23	



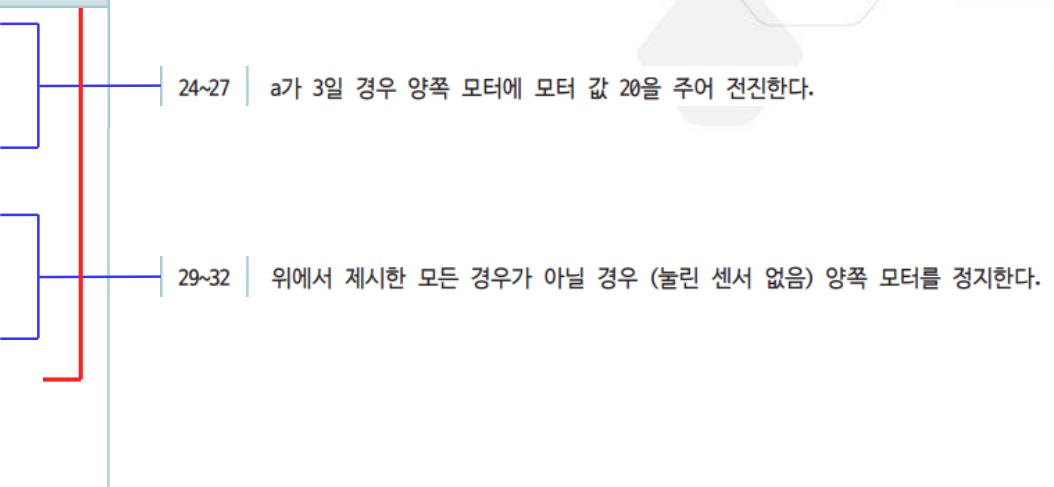
터치 센서를 이용한 주행

2-15

터치 센서를 이용한 주행(switch~case문)

```

24 case 3:
25     setMotorSpeed(lm, 20);
26     setMotorSpeed(rm, 20);
27     break;
28
29 default:
30     setMotorSpeed(lm, 0);
31     setMotorSpeed(rm, 0);
32     break;
33 }
34 }
35 }
```



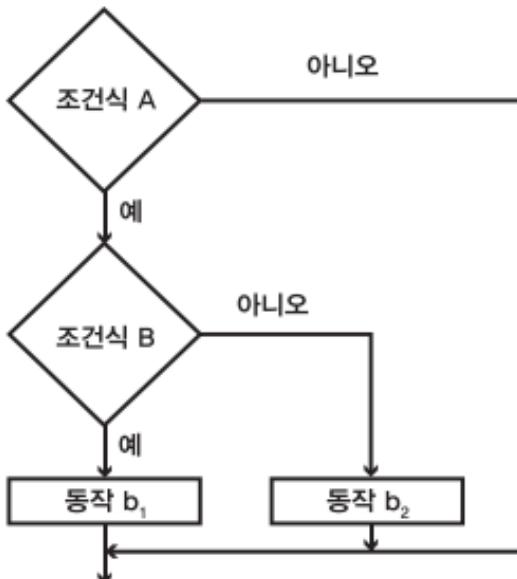
switch~case문과 if~else문의 비교

- if~else문은 어떠한 변수든지 비교 가능한 반면,
switch~case문은 정수형 변수만 비교 가능
- if~else문은 '범위의 비교'까지도 가능하지만
switch~case문은 '값의 비교'만 가능
- switch~case문의 경우 상황을 나누는 가짓수가 많아질수록 가독성이 좋음

복합조건 선택/실행

- 조건문을 복합적으로 사용하여 복잡한 문제 상황 제어 가능

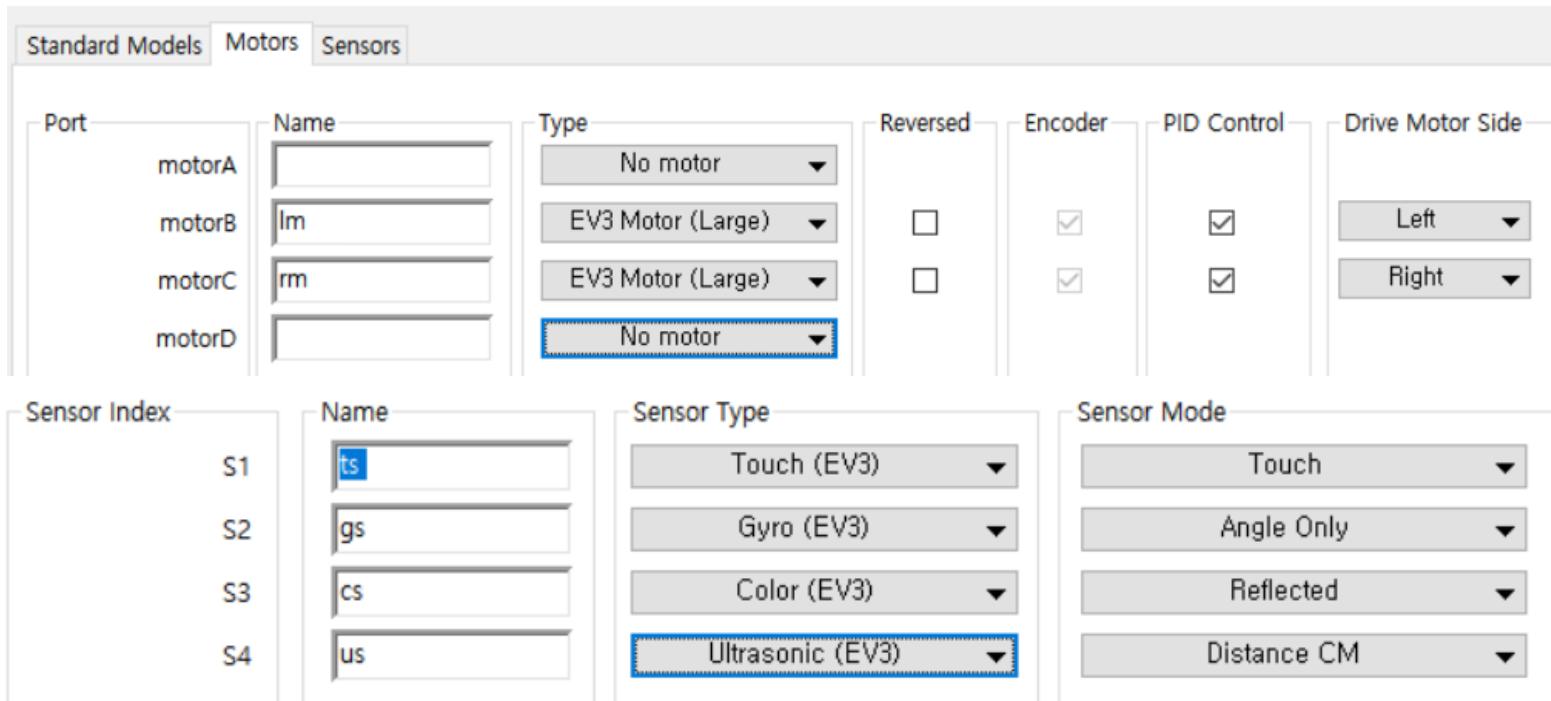
```
if(조건식 A)
{
    if(조건식 B)
    {
        동작 b1;
    }
    else
    {
        동작 b2;
    }
}
```



모터 및 센서 설정

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

Motors and Sensors Setup



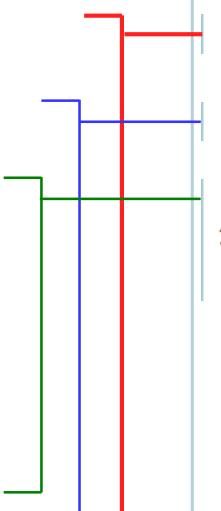
복합조건 선택/실행

2-16

사운드 출력과 주행(복합조건 선택/실행)

```

01 #pragma config(Sensor, S1, ts, sensorEV3_Touch)
02 #pragma config(Sensor, S3, cs, sensorEV3_Color)
03 #pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
04 #pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
05
06 task main()
07 {
08     while(1)
09     {
10         if(getTouchValue(ts) == 1)
11         {
12             if(getColorReflected(cs) < 30)
13             {
14                 setMotorSpeed(lm, 30);
15                 setMotorSpeed(rm, -30);
16                 sleep(2000);
17                 setMotorSpeed(lm, 0);
18                 setMotorSpeed(rm, 0);
19                 playSound(soundBeepBeep);
20             }
21         }
22     }
23 }
```



- 08 무한 반복(08번~34번 줄) 한다. while문은 이후의 장에서 다루도록 한다.
- 10 만약 터치 센서가 눌려있을 경우 아래 명령(11~33)을 수행한다.
- 12 만약 컬러 센서 값이 30 미만일 경우 아래 명령(13~20)을 수행한다.
- 14~18 회전을 2초간 하다가 정지한다.
- 19 BeepBeep 사운드를 재생한다.

복합조건 선택/실행

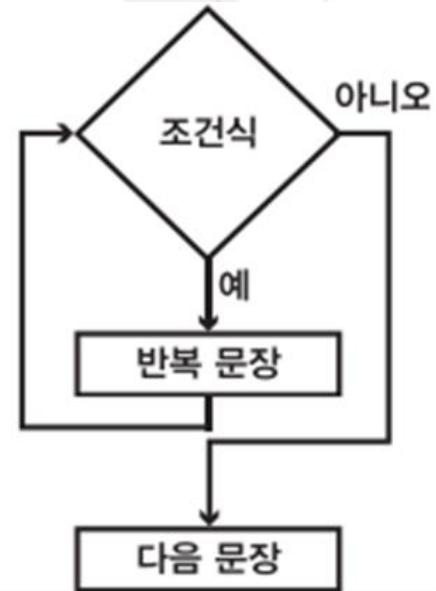
2-16	사운드 출력과 주행(복합조건 선택/실행)	
21	else	
22	{	
23	setMotorSpeed(lm, 30);	
24	setMotorSpeed(rm, 30);	
25	sleep(1000);	
26	setMotorSpeed(lm, -30);	
27	setMotorSpeed(rm, -30);	
28	sleep(1000);	
29	setMotorSpeed(lm, 0);	
30	setMotorSpeed(rm, 0);	
31	playSound(soundBlip);	
32	}	
33	}	
34	}	
35	}	

21 만약 컬러 센서 값이 30 이상일 경우 아래 명령(22~32)을 수행한다.
(12번 줄이 거짓일 경우 실행한다.)
23~30 전진과 후진을 1초씩 하고 정지한다.
31 Blip 사운드를 재생한다.

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; }
```



터치 센서를 이용한 반복 전진

2-17	터치 센서를 이용한 반복 전진(while문)
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	<u>while(getTouchValue(ts) == 0){ };</u>
08	setMotorSpeed(lm, 50);
09	setMotorSpeed(rm, 50);
10	sleep(500);
11	}
줄번호	코드 설명
07	터치 센서의 값이 0일 때(누르지 않았을 때) 로봇은 정지해 있다. 터치 센서를 누르면 센서 값이 1이 되면서 while문을 빠져나온다.
08~10	0.5초 동안 전진한다.



5회 반복 회전 후 사운드 출력

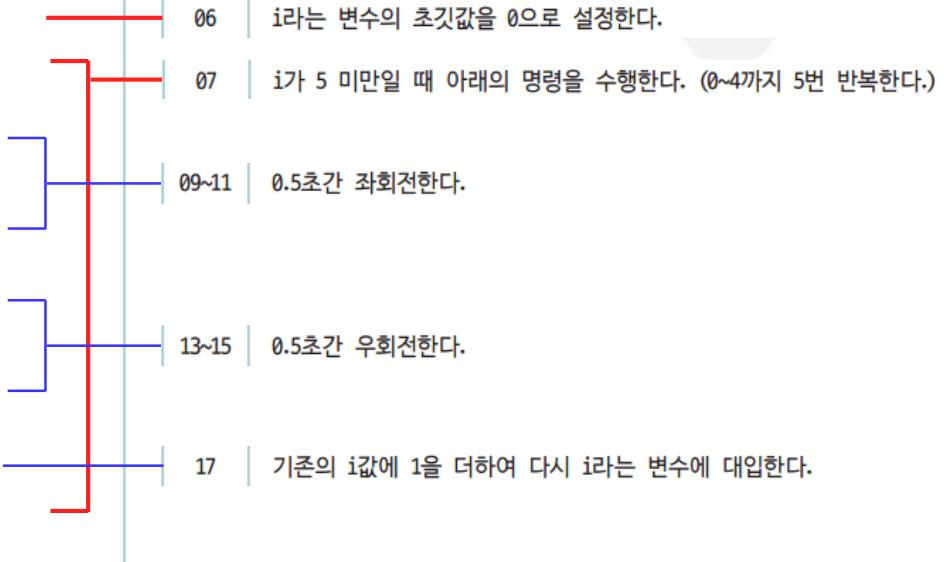
2-18

반복 회전 후 사운드 출력(while문)

```

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     int i = 0;
07     while(i < 5)
08     {
09         setMotorSpeed(lm, -50);
10         setMotorSpeed(rm, 50);
11         sleep(500);
12
13         setMotorSpeed(lm, 50);
14         setMotorSpeed(rm, -50);
15         sleep(500);
16
17         i++;
18     }

```



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 );
```



5회 반복 회전 후 사운드 출력

2-19

반복 회전 후 사운드 출력(do~while문)

```

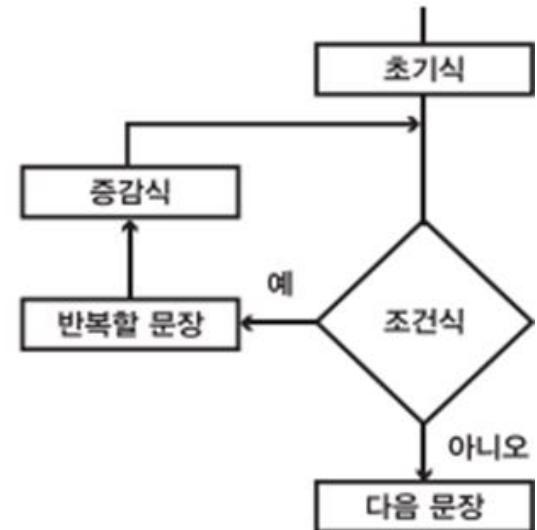
01 #pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
02 #pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
03
04 task main()
05 {
06     int i = 0;
07     do
08     {
09         setMotorSpeed(lm, 0);
10         setMotorSpeed(rm, 50);
11         sleep(500);
12
13         setMotorSpeed(lm, 50);
14         setMotorSpeed(rm, 0);
15         sleep(500);
16
17         i++;
18     }
19     while(i < 5);
20
21     setMotorSpeed(lm, 0);
22     setMotorSpeed(rm, 0);
23     playSound(soundBeepBeep);
24     sleep(500);
25 }
```



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; }
```



10회 회전 후 사운드 출력

2-20

일정 횟수만큼 반복 회전(for문)

```

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     int i;
07
08     for(i = 0; i < 10; i++)
09     {
10         setMotorSpeed(lm, 30);
11         setMotorSpeed(rm, 50);
12         sleep(500);
13
14         setMotorSpeed(lm, 50);
15         setMotorSpeed(rm, 30);
16         sleep(500);
17     }
18
19     setMotorSpeed(lm, 0);
20     setMotorSpeed(rm, 0);
21     playSound(soundBeepBeep);
22     sleep(500);
23 }
```

06 정수형 변수 i를 선언한다.

08 i가 0부터 9까지 아래의 반복문을 수행하면서 i++을 수행한다.

10~12 0.5초간 좌회전한다.

14~16 0.5초간 우회전한다.

19~20 동작 후 로봇을 정지시킨다.

21~22 0.5초간 사운드를 출력한다.

중첩 반복문

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

터치 센서와 컬러 센서를 이용한 주행

2-21	터치 센서와 컬러 센서를 이용한 주행(중첩 반복문)
01	#pragma config(Sensor, S1, ts, sensorEV3_Touch)
02	#pragma config(Sensor, S3, cs, sensorEV3_Color)
03	#pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
04	#pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
05	
06	task main()
07	{
08	while(getTouchValue(ts) == 0);
09	<u> while(getTouchValue(ts) == 1)</u>
10	{
11	setMotorSpeed(lm, 0);
12	setMotorSpeed(rm, 0);
13	<u> while(getColorReflected(cs) > 40)</u>
14	{
15	setMotorSpeed(lm, 50);
16	setMotorSpeed(rm, 50);
17	}
18	}
19	}
줄번호	코드 설명
08	터치 센서의 값이 0일 때(터치 센서가 눌리지 않았을 때)는 대기한다.
09	터치 센서의 값이 1일 때(터치 센서가 눌렸을 때) 로봇의 모터를 정지시킨다.
11~12	
13~17	컬러 센서 값이 40보다 클 때 (밝을 때) 계속해서 전진한다.



컬러 센서를 이용한 주행

2-12

컬러 센서를 이용한 주행(if문)

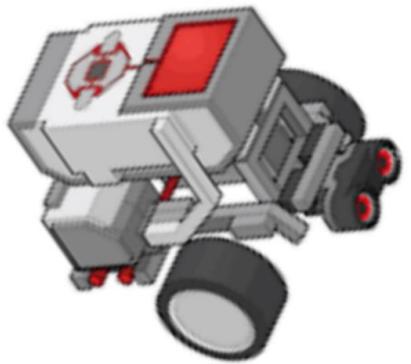
```

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.

RobotC project-1

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



EVnn_MilDang.c
로 저장.



Report

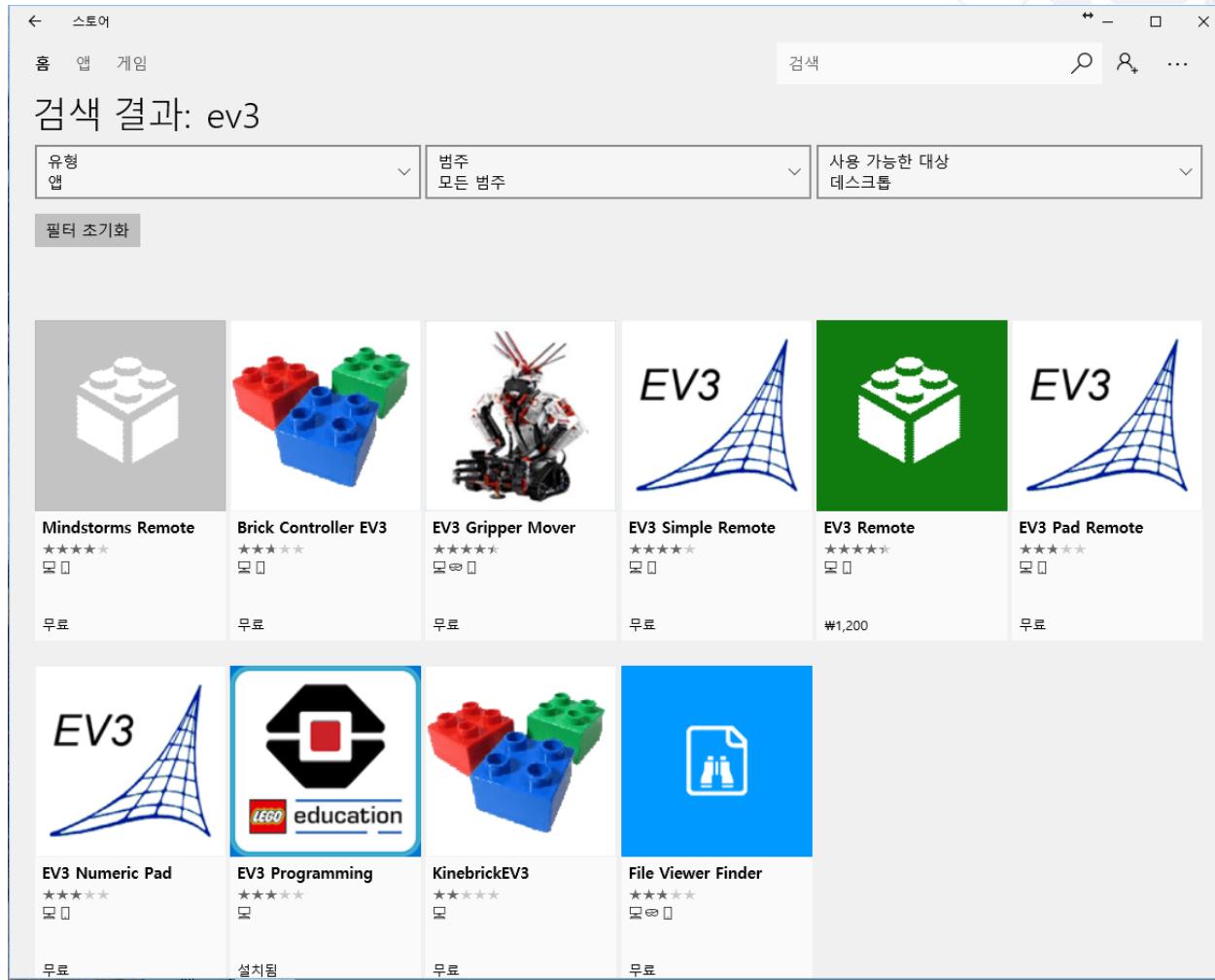
제출파일명 : EVnn_wk12.zip

- 압축할 파일들

- ① EVnn_team_info.c
- ② EVnn_MilDang.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.

또한 EV3 프로그래밍 앱에는 다양한 보조 자료가 포함되어 있어 교사와 학생 모두 즐거운 분위기 속에서 시작 단계를 시작할 수 있습니다. 여섯 가지의 단계별 로봇 에듀케이터 자습서는 프로그래밍과 하드웨어에 대한 효과적인 가이드를 제공합니다. 또한 로봇 수업 계획 소개 자료를 통해 교사들에게 아홉 가지의 초기 수업 개요를 제공하는 동시에 현지의 교육과정 표준과 연계된 새로운 학습 자료와 실행 가능한 평가 영역을 제시합니다...

[자세히](#)
- 제공 플랫폼**: PC
- 스크린샷**: Three screenshots showing the app's user interface, including a robot on a table and a programming block editor.
- 모두 표시** (Show all)



education

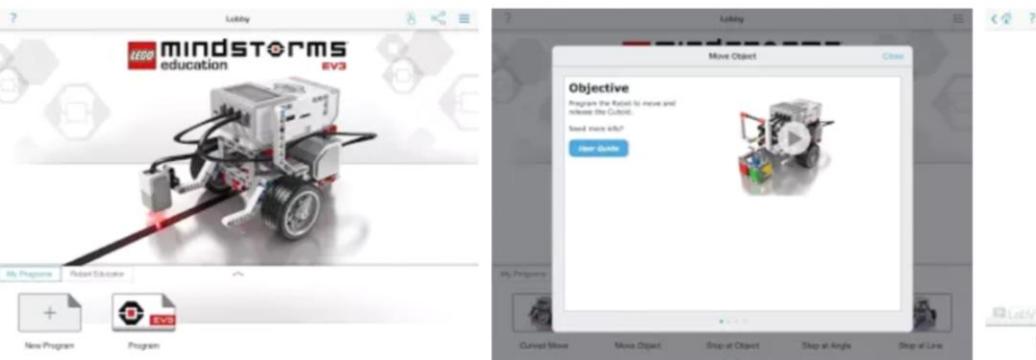
창의공학교육의 멘토

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 the app store page for the LEGO MINDSTORMS Education EV3 Programming app on an iPad. The page features the app's icon (a black and red geometric logo with the word "education" below it), its title "LEGO® MINDSTORMS® Education EV3", the developer "LEGO Education", and a rating of 4 stars from 6 reviews. A large blue "OPEN" button is prominent. Below the main section are three tabs: "Details", "Reviews", and "Related". Under the "Details" tab, there are two photographs illustrating the app's use: one showing two students working on a robot at a desk, and another showing three students sitting on the floor programming a robot with a tablet.

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