

로봇활용 SW교육 지침서

# The NEXT ROBOT with EV3

EV3로 배우는 블록 코딩 & C언어

2017년 2학기

인제대학교 헬스케어IT 학과

이상훈



education

나눔바른고딕

HandsOn  
Technology

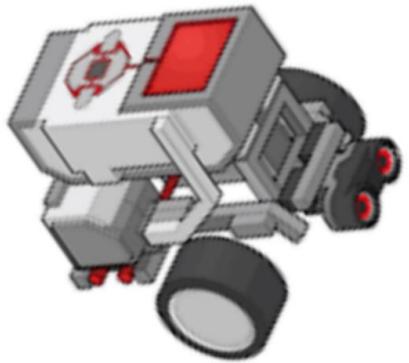
# Weekly plan (2<sup>nd</sup> semester, 2017)

- **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 : RobotC coding-3: Encoder & data logging**
- **wk15 : Final exam.**



# RobotC project-2

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



EVnn\_MilDang2.c  
로 저장.



# Report : best code (MilDang2)

```
9 #define EDGE_VALUE 30
10
11 task main()
12 {
13     int speed_l, speed_r, curval;
14     int initval = getGyroDegrees(gs);
15
16     // Gyro driving
17     while(getColorReflected(cs)>EDGE_VALUE)
18     {
19         curval = getGyroDegrees(gs) - initval;
20         speed_l = 15 - curval;
21         speed_r = 15 + curval;
22         setMotorSpeed(lm, speed_l);
23         setMotorSpeed(rm, speed_r);
24     }
}
```

```
26     // stop
27     if(getColorReflected(cs) <= EDGE_VALUE)
28     {
29         setMotorSpeed(lm, 0);
30         setMotorSpeed(rm, 0);
31         sleep(2000);
32     }
33     setMotorSpeed(lm, 50);
34     setMotorSpeed(rm, 50);
35
36     // MilDang
37     while(1) {
38         if(getUSDistance(us)<=10) {
39             setMotorSpeed(lm, -50);
40             setMotorSpeed(rm, -50);
41         }
42         else {
43             setMotorSpeed(lm, 50);
44             setMotorSpeed(rm, 50);
45         }
46     }
47 }
```

# Report : best code (MilDang2A)

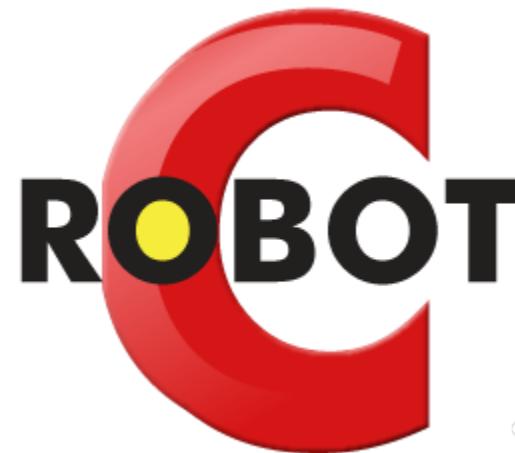
```
9 #define EDGE_VALUE 30
10
11 task main()
12 {
13     int speed_l, speed_r, curval;
14     int initval = getGyroDegrees(gs);
15
16     while(1)
17     { // Gyro driving
18         if(getColorReflected(cs)>EDGE_VALUE)
19         {
20             curval = getGyroDegrees(gs) - initval;
21             speed_l = 25 - curval;
22             speed_r = 25 + curval;
23             setMotorSpeed(lm, speed_l);
24             setMotorSpeed(rm, speed_r);
25         }
26     }
27     else
28     { // stop
29         setMotorSpeed(lm,0);
30         setMotorSpeed(rm,0);
31         sleep(2000);
32
33         setMotorSpeed(lm,50);
34         setMotorSpeed(rm,50);
35
36         // MilDang
37         while(1) {
38             if(getUSDistance(us)<=10) {
39                 setMotorSpeed(lm,-50);
40                 setMotorSpeed(rm,-50);
41             }
42             else {
43                 setMotorSpeed(lm,50);
44                 setMotorSpeed(rm,50);
45             }
46         }
47     }
48 }
```

장의공학교육의 멘토



# wk14: RobotC III.

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



# 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

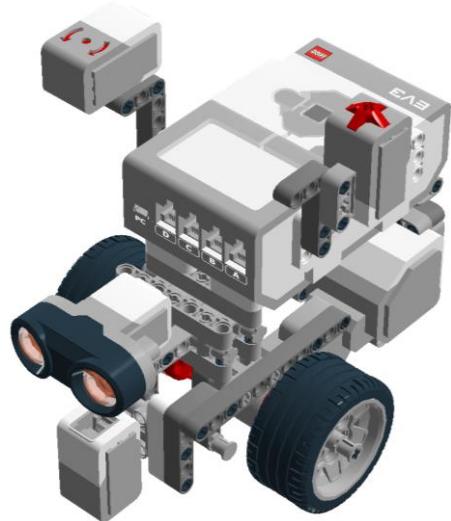


education

HandsOn  
Technology

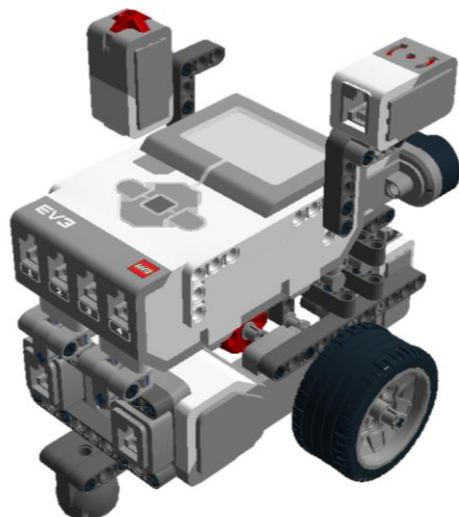
# 2. ROBOTC 기초

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



# 3. 액츄에이터 제어

- ◆ 모터
- ◆ 디스플레이
- ◆ 스피커
- ◆ LED



# 4. 센서 활용

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



# 엔코더 제어의 기초

- 라지 모터와 미디엄 모터는 회전한 각도를 알 수 있는 회전 센서(엔코더)를 내장하고 있어  $1^\circ$  단위로 제어 가능



라지 모터



미디엄 모터

# 엔코더 제어의 기초

- EV3 타이어의 지름을 이용하여 모터의 엔코더가 360°회전 시 이동한 거리를 알 수 있음



지름 = 5.6cm

원의 둘레 =  $2\pi r$  ( $r$  : 반지름) 의 공식에서 2r인 바퀴의 지름이 5.6cm이므로  
엔코더가 360도 회전했을 때 이동 거리는 약  $17.58(5.6 * 3.14)$ cm로 계산할 수 있다.

- 이 원리를 적용하여 임의의 거리를 직진하기 위해 회전해야 할 엔코더의 근사값을 구할 수 있음

$$360^\circ : 17.58\text{cm} = D^\circ : X\text{cm} \Rightarrow D = 360 * X / 17.58$$

# 엔코더 제어를 이용한 25cm 전진

## 4-6 엔코더 제어를 이용한 25cm 전진

```

01 #pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
02 #pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
03
04 int convert(float dist)
05 {
06     return (int)(360.0 * dist / 17.58);
07 }
08
09 task main()
10 {
11     int enc_degree;
12     enc_degree = convert(25.0);
13
14     resetMotorEncoder(lm);
15
16     while(getMotorEncoder(lm) < enc_degree)
17     {
18         setMotorSpeed(lm, 50);
19         setMotorSpeed(rm, 50);
20     }
21
22     setMotorSpeed(lm, 0);
23     setMotorSpeed(rm, 0);
24 }
```

04~07 이동할 거리를 실수형 변수 dist로 받아서 필요한 엔코더 값을 계산한 후 정수 형태로 변환하여 리턴한다.

12 convert 함수가 리턴한 값(엔코더 값)을 변수 enc\_degree에 저장한다.

14 왼쪽 모터의 엔코더 값을 0으로 초기화 한다.

16~20 왼쪽 모터의 엔코더 값이 변수 enc\_degree의 값보다 작은 동안 직진한다.

22~23 왼쪽 모터의 엔코더 값이 변수 enc\_degree의 값과 같아지면 정지한다.

**getMotorEncoder**

- Returns the encoder value of the motor plugged into the port specified by nMotorIndex in degrees.
  - \*The value returned is automatically converted from a float value to a whole number by ROBOTC

## Code Sample

```
//Reset the motor encoder value to zero
resetMotorEncoder(motorA);

//Set the target of motor A to 1000 encoder counts at speed 50
setMotorTarget(motorA, 1000, 50);

//Loop until the current encoder counts are equal to the target counts
repeatUntil(getMotorEncoder(motorA) == getMotorTarget(motorA))
{
}
```

**float getMotorEncoder(tMotor nMotorIndex)**

Parameter	Explanation	Data Type
Return Type	Function returns a *floating point decimal value	float
nMotorIndex	A motor port or name..	tMotor

# 엔코더 제어를 이용한 직진 및 후진

- 특정 모터의 엔코더 값이 목표 값이 될 때까지 모터를 회전시키고자 할 때에는 setMotorTarget() 함수를 사용
- 목표 엔코더 값은 절댓값으로 입력

# 엔코더 제어를 이용한 직진 및 후진

4-7

## 엔코더 제어를 이용한 직진 및 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     resetMotorEncoder(lm);
07     resetMotorEncoder(rm);
08     setMotorTarget(lm, 1000, 20);
09     setMotorTarget(rm, 1000, 20);
10     waitUntilMotorStop(lm);
11     resetMotorEncoder(lm);
12     resetMotorEncoder(rm);
13     setMotorTarget(lm, 1000, -20);
14     setMotorTarget(rm, 1000, -20);
15     waitUntilMotorStop(lm);
16 }
```

- |       |   |
|-------|---|
| 06~07 | 왼쪽 모터와 오른쪽 모터의 엔코더 값을 초기화한다.                            |
| 08~09 | 왼쪽 모터와 오른쪽 모터의 목표 엔코더 값을 1000으로 설정하고, 모터 값을 20으로 전진한다.  |
| 10    | 왼쪽 모터가 완전히 멈출 때까지 현재 상태를 유지한다.                          |
| 11~12 | 왼쪽 모터와 오른쪽 모터의 엔코더 값을 초기화한다.                            |
| 13~14 | 왼쪽 모터와 오른쪽 모터의 목표 엔코더 값을 1000으로 설정하고, 모터 값을 -20으로 후진한다. |
| 15    | 왼쪽 모터가 완전히 멈출 때까지 현재 상태를 유지한다.                          |

# ROBOTC

a C Programming Language for Robotics

## setMotorTarget

- This command tells the robot to move an absolute distance. It will add (or subtract) distance from the motor encoder's "0" position. The command needs three pieces of information
  - nMotorIndex:** The port or name of the motor to control.
  - nPosition:** The amount of degrees (absolute) that the motor should move
  - nSpeed:** The velocity of the motor.
- Setting either the target or the speed to a negative number will cause the motor to rotate backwards.
- At the end of this command, the motor will automatically come to a stop.

**void setMotorTarget(tMotor nMotorIndex, float nPosition, int nspeed)**

Parameter	Explanation	Data Type
Return Type	The function returns no value.	void
nMotorIndex	A motor port or name..	tMotor
nPosition	Target absolute value that the motor will travel to.	float
nSpeed	Speed value that will be passed to the motor (-100 to +100).	int



**Tip**

# 모터의 엔코더 값 이용하기

- 모터 값 50으로 1초간 직진하는 프로그램을 작성했을 때, 속도를 높이거나 낮추면 이동 거리가 늘어나거나 줄어듬
- 똑같은 거리를 이동하기 위해서는 sleep() 함수나 wait1Msec() 함수에 지정한 대기 시간을 변경해야 함
- 엔코더 값을 이용하면 이동 속도 및 대기 시간과 상관없이 항상 일정한 거리를 이동을 할 수 있음

# 모터 동기화 (Syncing motors)

- 모터 동기화란 하나의 모터를 기준으로 하여 두 개의 모터가 지정한 비율로 회전하도록 하는 것
- 모터 값을 기준으로 동기화 시 `setMotorSyncTime()` 함수를,  
엔코더 값을 기준으로 동기화 시 `setMotorSyncEncoder()` 함수를 사용

setMotorSyncTime(lm, rm, 100, 1000, 30)					
setMotorSyncTime()	lm	rm	100	1000	30
지정한 시간 동안 모터 동기화 실행	기준 모터	다른 모터	기준 모터 출력 비율	지정 시간	모터 값

setMotorSyncEncoder(lm, rm, 100, 1000, 30)					
setMotorSyncEncoder()	lm	rm	100	1000	30
지정한 엔코더 값에 도달할 때까지 모터 동기화 실행	기준 모터	다른 모터	기준 모터 출력 비율	지정 엔코더 값	모터 값

# ROBOTC

a C Programming Language for Robotics

## setMotorSyncTime

- Synchronizes two motors for a specified amount of time.
  - A value of 100 applies power to nMotorOne and negative power to nMotorTwo.
  - A value of -100 applies negative power to nMotorOne and power to nMotorTwo.
  - A value of 0 applies equal power to nMotorOne and nMotorTwo (straight movement).
  - A value of 50 applies all of the power to nMotorOne and zero power to nMotorTwo.
  - A value of -50 applies zero power to nMotorOne and all of the power to nMotorTwo.

`void setMotorSyncTime(tMotor nMotorOne, tMotor nMotorTwo, long nTurnRatio, long nTimeMsec, long nSignedPower)`

Parameter	Explanation	Data Type
Return Type	The function returns no value.	void
nMotorOne	Port number or name for the first motor.	tMotor
nMotorTwo	Port number or name for the second motor.	tMotor
nTurnRatio	Power level ratio between nMotorOne and nMotorTwo.	long
nTimeMsec	Time (in milliseconds) to run nMotorOne and nMotorTwo for.	long
nSignedPower	Base power level used by nTurnRatio to apply individual power levels to the motors	long



# 모터 동기화를 이용한 직진

4-8	모터 동기화를 이용한 직진
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>setMotorSyncTime(lM, rM, 0, 3000, 30);</u>
07	sleep(3000);
08	}
줄번호	코드 설명
06	왼쪽 모터를 기준 모터로 하여 3초 동안 오른쪽 모터를 왼쪽 모터와 동일한 모터 값 30으로 지정한다.
07	현재 상태를 3초간 유지한다.

# 모터 동기화를 이용한 스윙 턴

4-9	모터 동기화를 이용한 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>setMotorSyncEncoder(lM, rM, 100, 1000, 30);</u>
07	sleep(5000);
08	}
줄번호	코드 설명
06	왼쪽 모터를 기준 모터로 하여 왼쪽 모터의 엔코더 값이 10000이 될 때까지 왼쪽 모터에만 모터 값 30을 지정한다.
07	현재 상태를 5초간 유지한다.

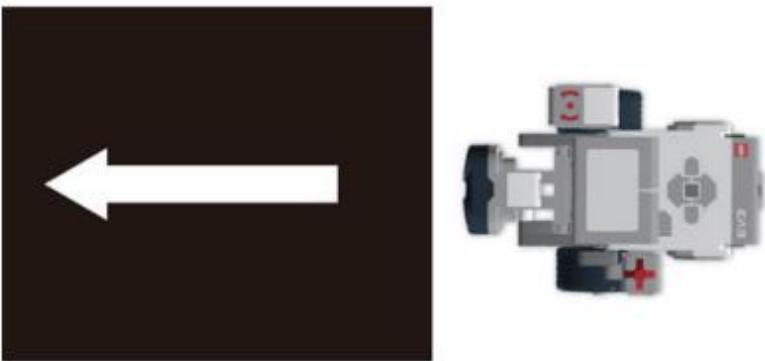
**Tip**

# 시간 함수 사용

- setMotorSyncTime() 함수나 setMotorSyncEncoder() 함수는 그 자체로 모터가 동작하지 않음
- **동작을 유지하는 시간을 설정하기 위한 함수인 sleep()이 함께 와야 함**

# 타이머를 이용한 주행 시간 출력

- 로봇이 검정 구간을 빠져나가는데 걸린 시간을  
LCD 디스플레이 화면에 나타내는 소스 코드



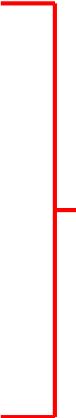
# 타이머를 이용한 주행 시간 출력

4-10	타이머를 이용한 주행 시간 출력
------	-------------------

```

01 #pragma config(Sensor, S3, cs, sensorEV3_Color, modeEV3Color_Color)
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     const int black = 1;
08     int speed = 30;
09     float t;
10
11     while(getColorName(cs) != black)
12     {
13         setMotorSpeed(lm, speed);
14         setMotorSpeed(rm, speed);
15     }
16     clearTimer(T1);

```



- |       |   |
|-------|---|
| 07    | 검정색 라인의 값을 1로 설정한다. (중간에 변경되지 않도록 상수로 선언)                   |
| 08    | 로봇의 주행 속도를 변수 speed에 지정하고 이를 변경하면서 속도에 따른 시간을 확인할 수 있도록 한다. |
| 09    | 주행 시간을 저장하기 위한 변수 t를 실수형으로 지정한다.                            |
| 11~15 | 검정색을 만날 때까지 직진한다.   |

Color sensor → Color mode

S3 | cs | Color (EV3) | Color

# 타이머를 이용한 주행 시간 출력

4-10 타이머를 이용한 주행 시간 출력

```

18 while(getColorName(cs) == black)
19 {
20     setMotorSpeed(lm, speed);
21     setMotorSpeed(rm, speed);
22 }
23 t = time1[T1];
24 displayBigTextLine(1, "record = %.2f sec", t/1000);
25
26 setMotorSpeed(lm, 0);
27 setMotorSpeed(rm, 0);
28 sleep(10000);
29 }
```

18-22	검정색을 인식하는 동안 직진한다.
23	흰색을 만나는 순간까지 걸린 시간을 변수 t에 1/1000초 단위로 저장한다.
24	변수 t의 값을 초 단위로 변경하여 LCD 디스플레이에 출력한다.
26~28	로봇을 멈추고 LCD 디스플레이에 출력한 상태로 10초간 대기한다.

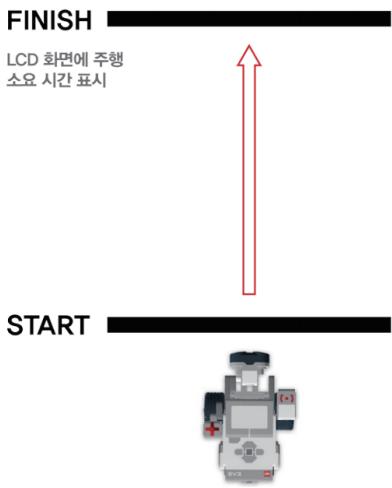
**Tip**

# 타이머

- 타이머는 **T1~T4**까지 4개를 사용 가능
- 타이머 값은 **time1 []**을 이용하여 1/1000초 단위로 측정할 수 있음
- **time10 []**은 1/100초, **time100 []**은 1/10초 단위로 타이머 값을 측정할 때 이용

# 주행 시간 맞추기

- 주행 시간은 START 라인을 통과하는 시간부터 FINISH 라인에 닿는 순간까지 측정하여 주행 종료 후 LCD 화면에 출력
- 로봇의 속도와 라인 사이의 거리를 알려주고, 로봇이 라인 사이를 통과하는 데 걸리는 시간을 미리 계산한 후 출력된 시간과 비교하는 게임 ( $T \sim 3.8 \text{ sec}$ )



**Tip**

# 로봇의 속력

- 엔코더 제어 단원에서 정확한 거리 직진을 위해 사용했던 다음의 식을 응용함  
( 바퀴의 지름이 5.6cm임을 이용)

$$360^\circ : 17.58\text{cm} = D^\circ : X\text{cm} \Rightarrow D = 360 * X / 17.58$$

- 로봇의 속력을 계산하기 위한 단위로는 거리/시간(m/sec, cm/sec), 회전각도/시간(degree/sec) 등을 사용할 수 있음



- 모터 값을 20부터 80까지로 한 이유는 모터 값이 너무 작거나 또는 너무 큰 경우에는 다양한 외적 요인들로 인해 모터 값과 엔코더 값 사이의 관계가 일정하지 않기 때문이며, 따라서 **여러 번 실험을 해서 얻은 평균 값을** 이용하게 된다

# 주행 시간 맞추기

4-14

## 주행 시간 맞추기

```

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 task main()
06 {
07     const int black = 1;
08     int speed = 30;
09     float t;
10
11     while(getColorName(cs) != black)
12     {
13         setMotorSpeed(lm, speed);
14         setMotorSpeed(rm, speed);
15     }
16     while(getColorName(cs) == black)
17     {
18         setMotorSpeed(lm, speed);
19         setMotorSpeed(rm, speed);
20     }

```

09

주행 시간을 저장하기 위한 변수 t를 실수형으로 지정한다.

11~15

검정색을 만날 때까지 직진한다.

16~20

검정색을 벗어날 때까지 직진한다.

# 주행 시간 맞추기

4-14	주행 시간 맞추기
22	clearTimer(T1);
23	
24	setMotorSpeed(lm, speed);
25	setMotorSpeed(rm, speed);
26	sleep(100);
27	
28	while(getColorName(cs) != black)
29	{
30	setMotorSpeed(lm, speed);
31	setMotorSpeed(rm, speed);
32	}
33	t = time1[T1];
34	displayTextLine(1, "record = %.2f sec", t / 100);
35	
36	setMotorSpeed(lm, 0);
37	setMotorSpeed(rm, 0);
38	sleep(10000);
39	}

24~26 검정색을 확실하게 벗어나도록 0.1초간 직진한다.

28~32 검정색을 만날 때까지 직진한다.

33 타이머 T1의 초기화 이후 현재까지 걸린 시간을 변수 t에 저장한다.

34 변수 t의 값을 LCD 화면에 출력한다.

36~38 로봇을 멈추고 LCD 화면에 출력한 상태로 10초간 대기한다.

# 데이터 로깅 (Logging data)

# 데이터 로깅 (Logging data)

- 데이터 로깅 : **로봇 동작 중 특정 센서 값 및 모터 값의 변화를 기록**하는 방법
- 데이터 로깅 실행 방법은 2가지로, '**Program Debug**' 대화 창을 이용하는 방법과 **데이터 로깅 함수**를 이용하는 방법이 있음

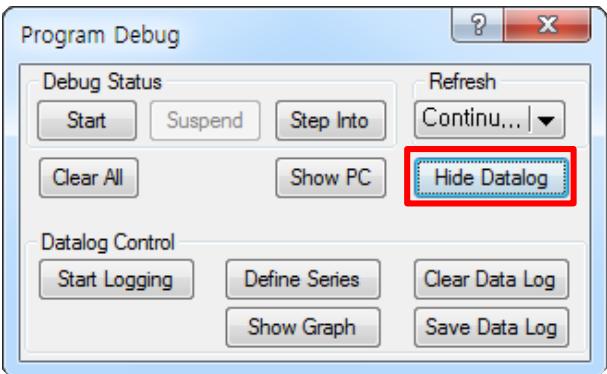
# Program Debug를 이용한 데이터 로깅

4-11	Program Debug를 이용한 데이터 로깅
01	#pragma config(Sensor, S1, ts, sensorEV3_Touch)
02	#pragma config(Sensor, S2, gs, sensorEV3_Gyro)
03	#pragma config(Sensor, S3, cs, sensorEV3_Color)
04	#pragma config(Sensor, S4, ss, sensorEV3_Ultrasonic)
05	#pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)
06	#pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)
07	
08	task main()
09	{
10	resetMotorEncoder(lm);
11	resetMotorEncoder(rm);
12	
13	while(getTouchValue(ts) == 0){ }
14	while(getTouchValue(ts) == 1){ }
15	
16	setMotorSpeed(lm, 30);
17	setMotorSpeed(rm, -30);
18	sleep(5000);
19	
20	setMotorSpeed(lm, -30);
21	setMotorSpeed(rm, 30);
22	sleep(5000);
23	}

- 10~11 양쪽 모터의 엔코더 값을 0으로 초기화시킨다.
- 13~14 터치 센서가 눌렀다 떼질 때까지 대기한다.
- 16~18 시계 방향으로 포인트 턴을 한다.
- 20~22 반시계 방향으로 포인트 턴을 한다.

# 데이터 로깅 - 'Program Debug' 대화창

- 코드 작성 후, F5(Compile and Download Program)를 누르면 'Program Debug' 대화 창이 나타남
- 이 창에서 [Show Datalog] 버튼을 누르면 'Datalog Control'에 관한 버튼이 나타남
- 이 버튼들을 사용해 기록된 데이터를 저장 및 그래프로 변환 가능



# 데이터 로깅 - 'Program Debug' 대화창

- 데이터 로깅을 하기 위해 가장 먼저 '데이터 정의'를 해야 함

데이터 정의 : 데이터의 종류, 성격 및 주기(Poll Rate)를 설정하는 것

- 대화 창에서 [Define Series] 버튼을 누르면 나타나는 'Datalog Control' 창 이용

Datalog Control

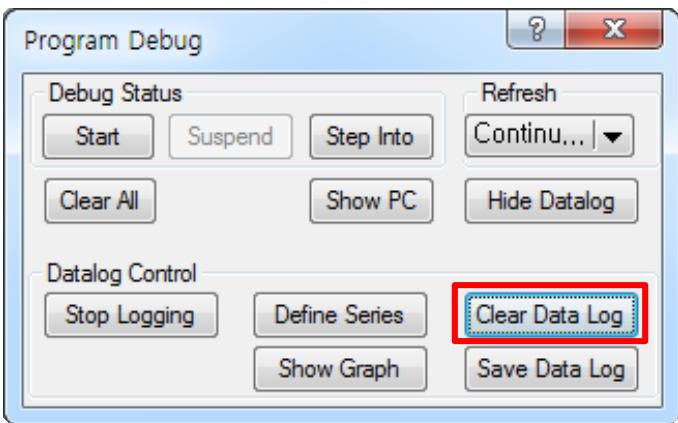
The dialog box contains a table with the following data:

Col...	Series Name	Source	Property Type	Index	Poll Rate
1	Series_0	Motors	MotorPower	motorB	100 MilliSeconds
2	Series_1	Motors	MotorPWM	motorC	100 MilliSeconds
3	Series_2	Sensors	Sensor	S1	100 MilliSeconds
4	Series_3	Sensors	Sensor	S2	100 MilliSeconds
5	Series_4	Sensors	Sensor	S3	100 MilliSeconds
6	Series_5	Sensors	Sensor	S4	100 MilliSeconds
7	Series_6				
8	Series_7				

Edit  
Delete  
Move Up  
Move Down

# 데이터 로깅 - 'Program Debug' 대화창

- 데이터 정의 후, 'Program Debug' 대화 창의 [Clear Data Log] 버튼을 눌러 기존 데이터 로깅 결과를 초기화함



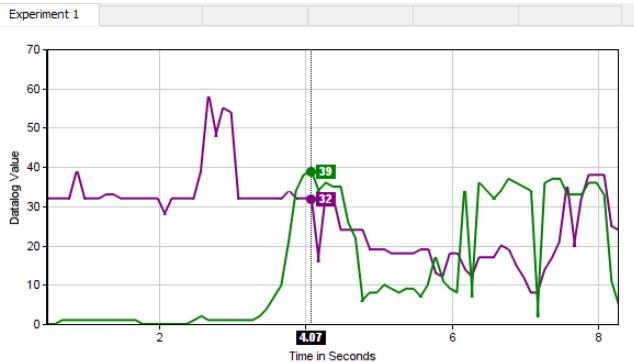
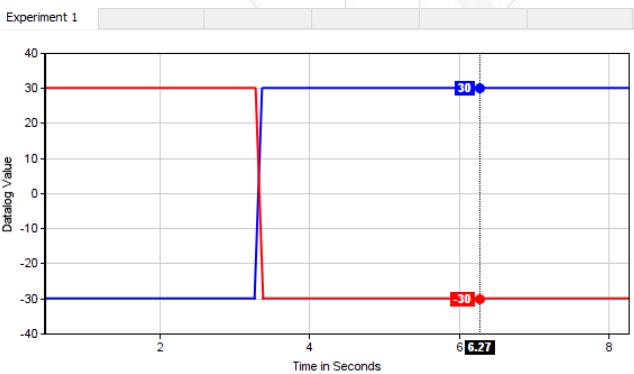
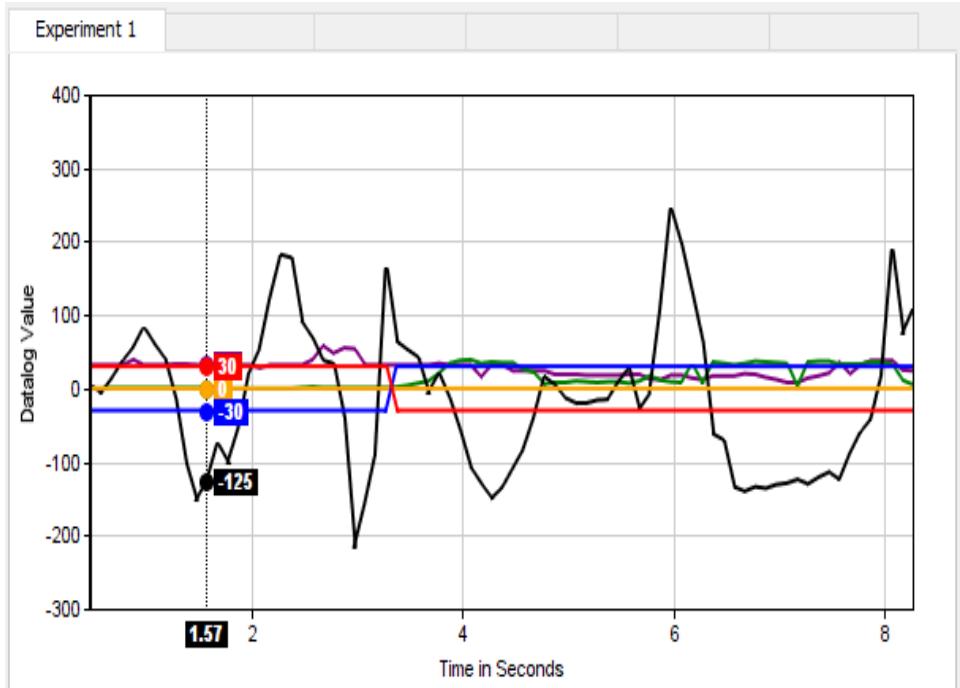
# 데이터 로깅 - 'Program Debug' 대화창

- [Start] 버튼을 눌러 로봇을 동작시킨 후, [Start Logging] 버튼을 눌러 데이터 로깅을 시작함
- 로봇의 동작이 멈추면 [Stop Logging] 버튼을 눌러 데이터 로깅 종료시킴



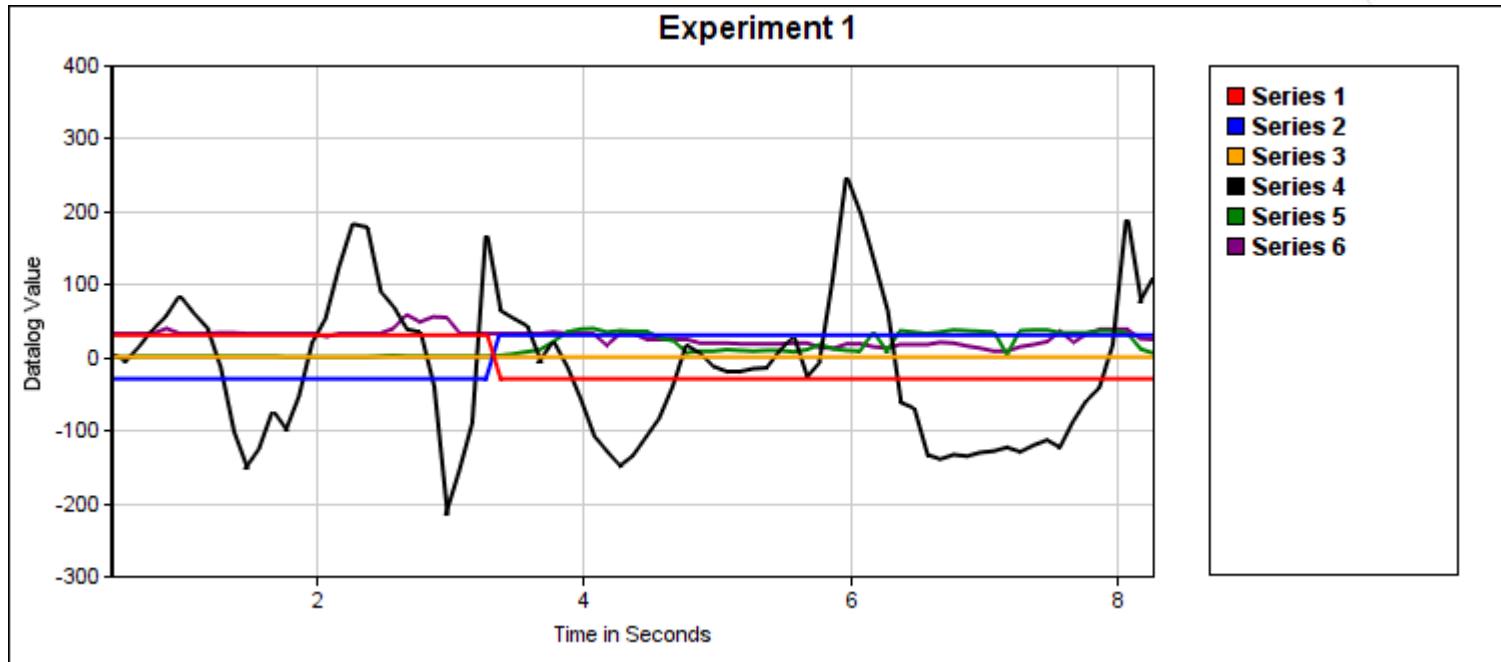
# 데이터 로깅 - 'Program Debug' 대화창

- [Show Graph] 버튼을 누르면 나타나는 'Datalog Graph' 대화 창에서 데이터 로깅 결과 확인



# 데이터 로깅 - 'Program Debug' 대화창

- 'Datalog Graph' 대화 창의 저장 버튼 이용 시 데이터 로깅 결과를 텍스트 파일(txt) 또는 이미지 파일(.png)로 저장 가능



→ 결과 저장 : [EVnn\\_RC\\_DL.png](#)

# 데이터 로깅 - 'Program Debug' 대화창

- 'Datalog Graph' 대화 창의 저장 버튼 이용 시 데이터 로깅 결과를 텍스트 파일(csv)로 저장 가능

A	B	C	D	E	F	G	H
Row	Timestamp	Series 1	Series 2	Series 3	Series 4	Series 5	Series 6
2	0	0.47	30	-30	0	3	0
3	1	0.57	30	-30	0	-5	0
4	2	0.67	30	-30	0	15	1
5	3	0.77	30	-30	0	38	1
6	4	0.87	30	-30	0	56	1
7	5	0.97	30	-30	0	84	1
8	6	1.07	30	-30	0	61	1
9	7	1.17	30	-30	0	41	1
10	8	1.27	30	-30	0	-13	1
11	9	1.37	30	-30	0	-101	1
12	10	1.47	30	-30	0	-151	1
13	11	1.57	30	-30	0	-125	1
14	12	1.67	30	-30	0	-74	1
15	13	1.77	30	-30	0	-100	0
16	14	1.87	30	-30	0	-53	0
17	15	1.97	30	-30	0	21	0
18	16	2.07	30	-30	0	54	0
19	17	2.17	30	-30	0	124	0
20	18	2.27	30	-30	0	182	0
21	19	2.37	30	-30	0	178	0
22	20	2.47	30	-30	0	91	1

→ 데이터 저장 :  
EVnn\_RC\_DL.csv

# 라이브러리 함수를 이용한 데이터 로깅

4-12	라이브러리 함수를 이용한 데이터 로깅
01	#pragma config(Sensor, S2, gs, sensorEV3_gs)
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 gyro_val;
08	
09	<u>datalogOpen(0, 1, false);</u>
10	
11	clearTimer(T1);
12	while(time1[T1] < 5000)
13	{
14	setMotorSpeed(lm, 30);
15	setMotorSpeed(rm, 30);
16	gyro_val = getGyroDegrees(gs);
17	datalogAddShort(0, gyro_val);
18	sleep(300);
19	}
20	
21	<u>datalogClose();</u>
22	}

07 자이로 센서의 값을 저장하기 위한 변수 gyro\_val을 선언한다.

09 데이터 로깅 저장 파일을 1컬럼 형식, 덮어쓰기 모드로 지정한다.

11 내장 타이머의 값을 0으로 초기화한다.

12~19 5초 동안 데이터를 수집한다.

14~15 양쪽 모터에 모터값 30을 전달한다.

16 변수 gyro\_val에 자이로 센서의 값을 저장한다.

17 변수 gyro\_val의 값을 데이터 로깅 저장 파일에 저장한다.

18 0.3초간 대기한다.

21 데이터 로깅 저장 파일을 닫는다.

**Tip**

# 데이터 로깅 함수

- `datalogOpen()` 함수의 인수는 파일 인덱스, 칼럼 수, 데이터 저장 모드 3개이다.

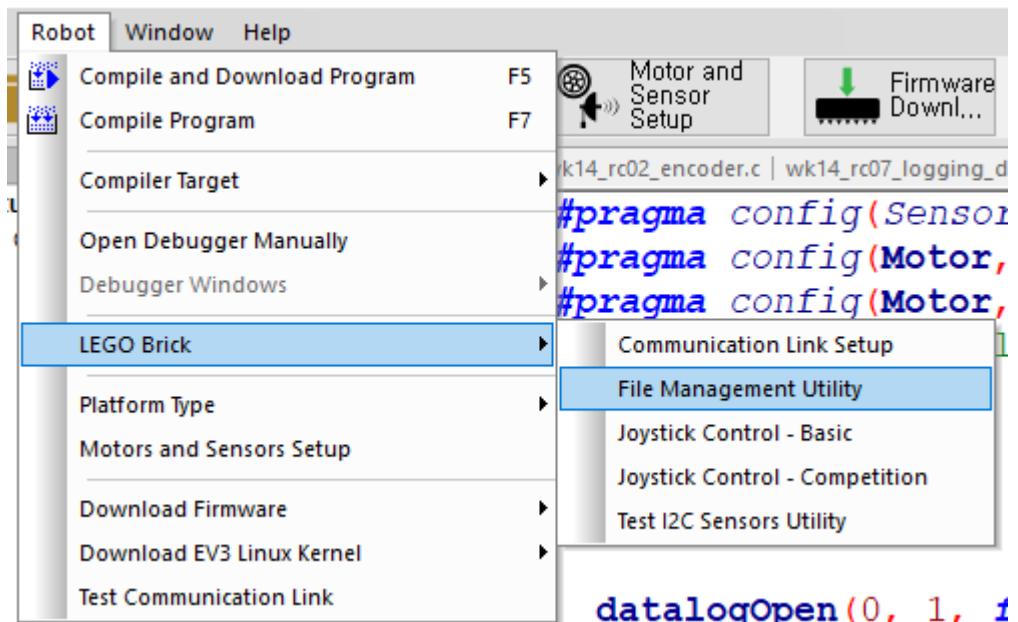
```
datalogOpen(0, 1, false); // 0번 파일의 1개의 컬럼에 새로 데이터를 기록  
datalogOpen(0, 1, true); // 0번 파일의 1개의 컬럼에 덧붙여서 데이터를 기록  
datalogAddShort(0, gyro_val); // 0번 컬럼에 gyro_val 값을 기록  
  
datalogOpen(1, 2, false); // 1번 파일의 2개의 컬럼에 새로 데이터를 기록  
datalogAddShort(0, col_val); // 0번 컬럼에 col_val 값을 기록  
datalogAddShort(1, gyro_val); // 1번 컬럼에 gyro_val 값을 기록
```

# 데이터 로깅 - 데이터 로깅 함수

- **데이터 로깅 함수를 이용한 데이터 로깅은 한 줄에 최대 4개 칼럼까지 지원**
  - 4종류 센서 값을 한 번에 저장 가능,
  - 칼럼 개수는 `datalogOpen()` 함수에서 지정
- 데이터를 파일에 저장할 때 데이터 종류에 따라
  - `datalogAddShort()` 함수,
  - `datalogAddLong()` 함수,
  - `datalogAddChar()` 함수 등 사용 가능

# 데이터 로깅 - 데이터 로깅 함수

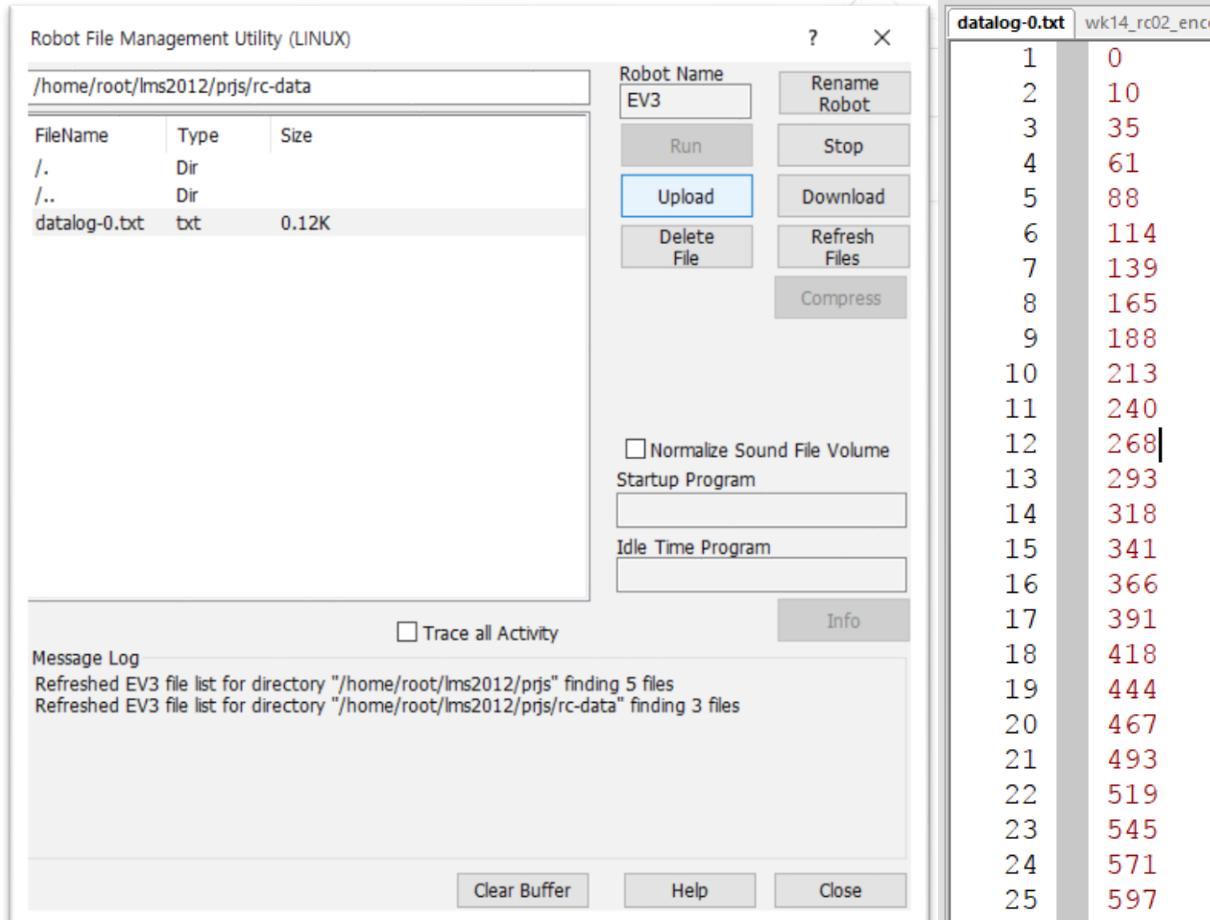
- 코드 실행으로 생성된 데이터 파일은 아래 그림과 같이  
[Robot] - [LEGO Brick] - [File Management Utility] 메뉴를 클릭해  
컴퓨터로 다운로드 가능



# 데이터 로깅 - 데이터 로깅 함수

- 데이터 로깅 저장 파일은 'datalog-0.txt'으로 파일명이 자동으로 부여
- [rc-data] 폴더 안에 저장됨
- 컴퓨터로 파일을 저장할 때에는  
[UPLOAD] 이용

→ 데이터 파일 확인 및 분석



# 라이브러리 함수를 이용한 데이터 로깅

4-13	라이브러리 함수를 이용한 데이터 로깅의 응용	
01	#pragma config(Sensor, S1, ts, sensorEV3_Touch)	
02	#pragma config(Sensor, S2, gs, sensorEV3_Gyro)	
03	#pragma config(Sensor, S3, cs, sensorEV3_Color)	
04	#pragma config(Sensor, S4, ss, sensorEV3_Ultrasonic)	
05	#pragma config(Motor, motorB, lm, tmotorEV3_Large, PIDControl, encoder)	
06	#pragma config(Motor, motorC, rm, tmotorEV3_Large, PIDControl, encoder)	
07		
08	task main()	
09	{	
10	datalogOpen(0, 4, false);	10     데이터 로깅 저장 파일을 4컬럼 형식, 덮어쓰기 모드로 지정한다.
11		
12	clearTimer(T1);	12     타이머 T1을 초기화하여 시간 측정을 시작한다.
13	while(time1[T1] < 5000)	
14	{	
15	setMotorSpeed(lm, 30);	13~22     5초 동안 데이터를 수집한다.
16	setMotorSpeed(rm, -30);	15~16     로봇이 시계 방향으로 포인트 턴하도록 한다.
17	datalogAddShort(0, getTouchValue(ts));	17     0번 컬럼에 터치 센서 값을 기록한다.
18	datalogAddShort(1, getGyroDegrees(gs));	18     1번 컬럼에 자이로 센서 값을 기록한다.
19	datalogAddShort(2, getColorReflected(cs));	19     2번 컬럼에 컬러 센서 값을 기록한다.
20	datalogAddShort(3, getUSDistance(ss));	20     3번 컬럼에 초음파 센서 값을 기록한다.
21	sleep(200);	
22	}	
23		
24	setMotorSpeed(lm, 0);	24~25     로봇을 정지시킨다.
25	setMotorSpeed(rm, 0);	
26	datalogClose();	26     데이터 로깅 저장 파일을 닫는다.
27	}	

# 데이터 로깅 - 데이터 로깅 함수 응용

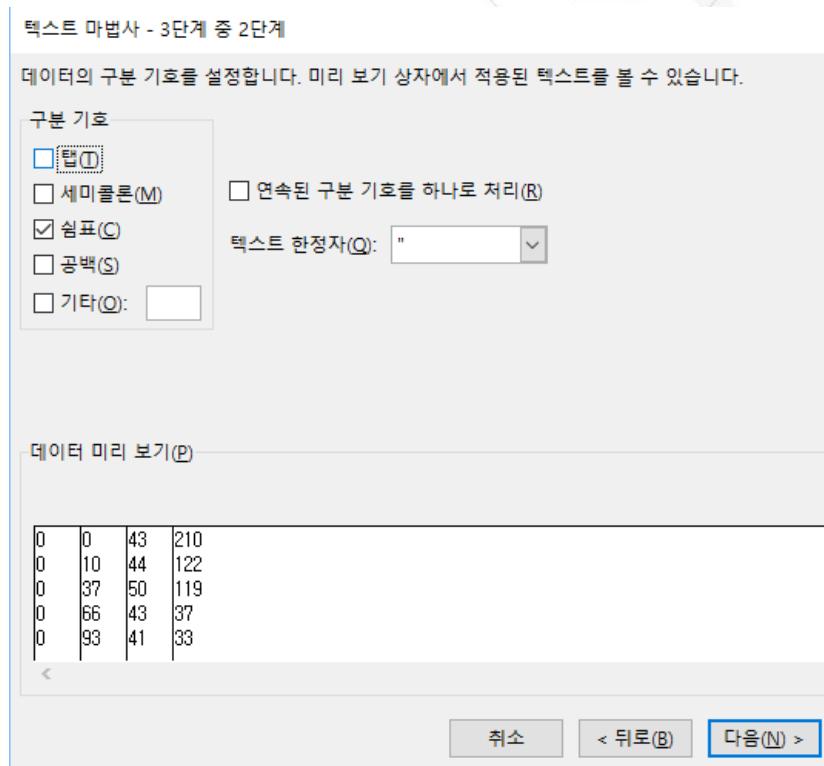
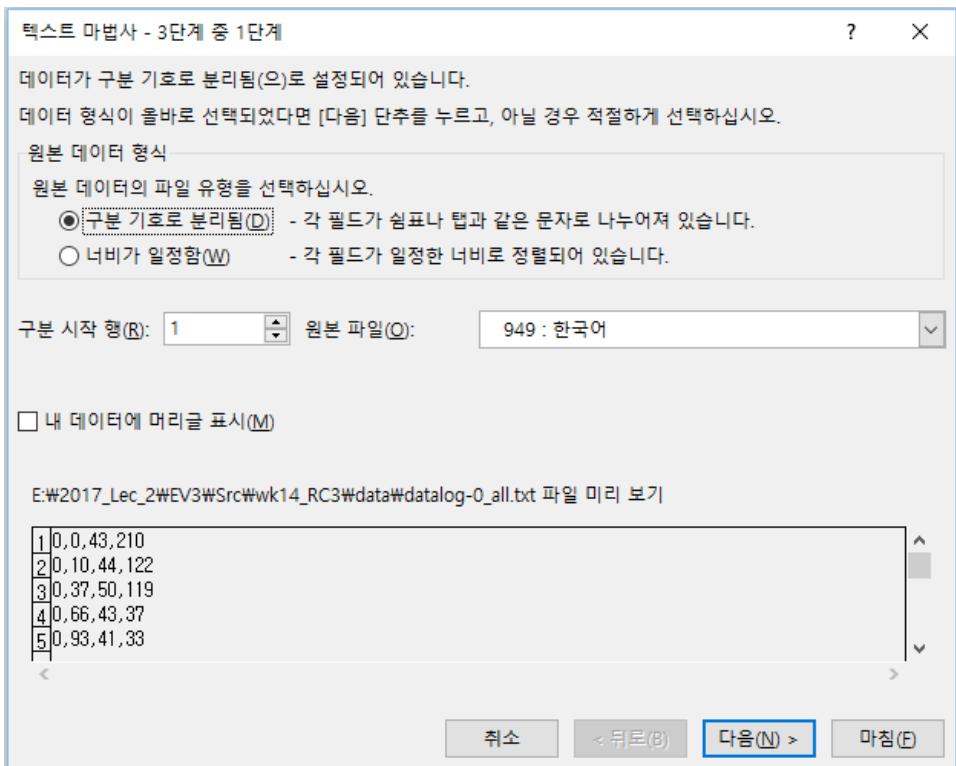
- 각종 센서 값이 기록된 'data-0.txt' 파일을 이용하여 스프레드시트로 불러와 그래프를 그릴 수 있음
  1. Ts
  2. Gs
  3. Cs
  4. Us

```

1 0,0,43,210
2 0,10,44,122
3 0,37,50,119
4 0,66,43,37
5 0,93,41,33
6 0,118,42,34
7 0,142,49,32
8 0,168,45,34
9 0,194,46,32
10 0,224,42,35
11 0,253,43,46
12 0,280,49,38
13 0,304,47,49
14 0,328,42,64
15 0,354,45,137
16 0,382,45,119
17 0,410,44,118
18 0,436,43,36
19 0,460,42,34
20 0,485,40,32
21 0,511,44,29
22 0,537,46,34
23 0,564,45,35
24 0,590,44,33
25 0,614,42,45
  
```

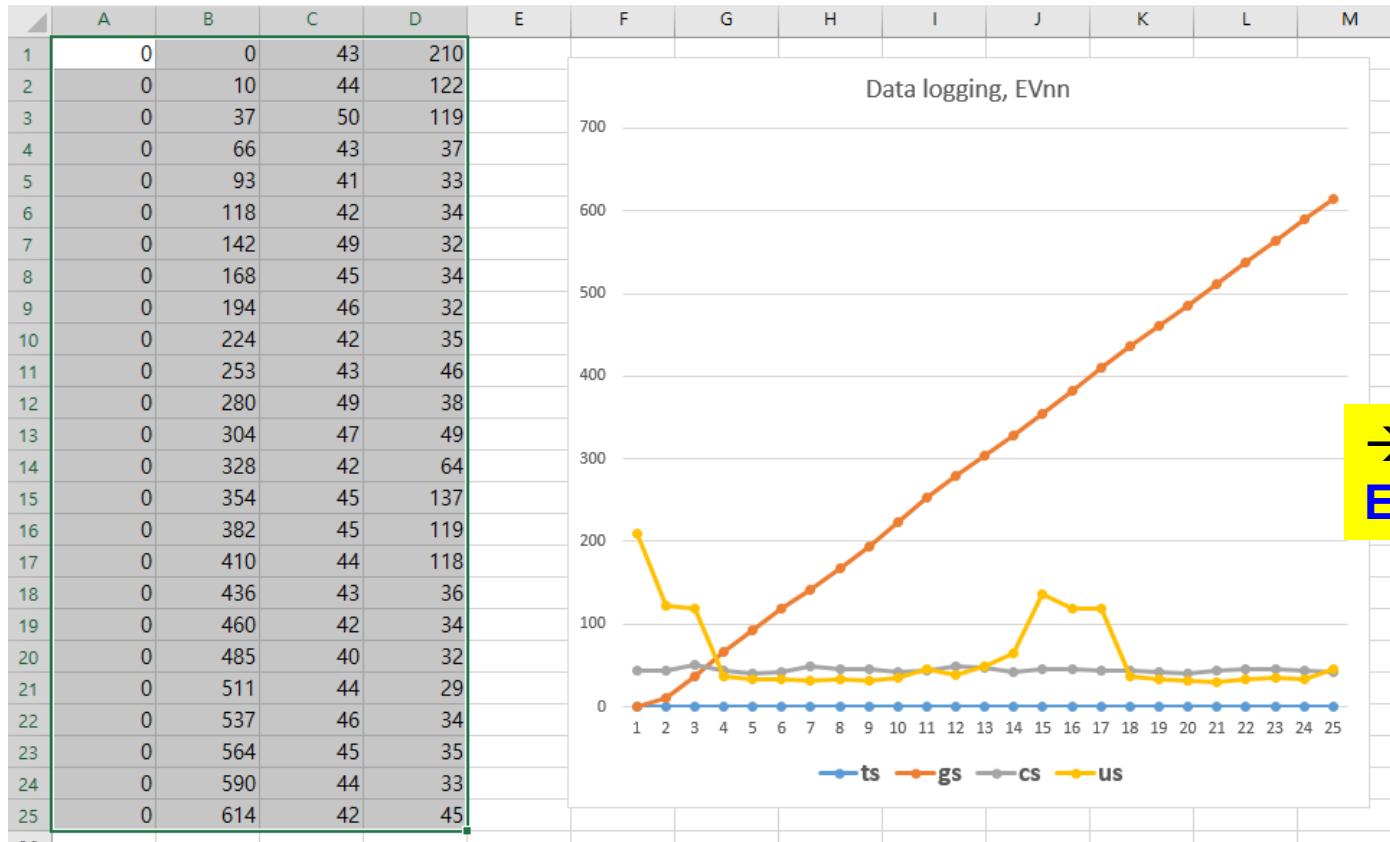
# 데이터 로깅 - 데이터 로깅 함수 응용

- 스프레드시트를 통해 'datalog-0.txt'를 열면 '텍스트 마법사' 대화 창이 나타남
- 다음의 그림처럼 파일이 쉼표(,)로 구분되어 있음을 고려해 파일을 불러옴



# 데이터 로깅 - 데이터 로깅 함수 응용

- 스프레드시트로 불러온 데이터를 가공하여 새로운 정보(평균, 순위 등)를 추출할 수 있음
- 시간에 따른 센서 값의 변화 과정 확인할 수 있음



→ 데이터 저장 :  
EVnn\_RC\_DL2.xls

# 데이터 로깅 - 데이터 로깅 함수 응용 II

- ▼  Command Library - LEGO EV3
  - >  Graphical
  - >  Natural Language
  - ▼  ROBOTC
    - >  Battery and Power Control
    - >  Buttons
    - ▼  Datalogging
      -  `datalogAddValue`
      -  `datalogAddValueWithTimeStamp`
      -  `datalogBackgroundPollingPause`
      -  `datalogBackgroundPollingResume`
      -  `datalogClear`
      -  `datalogDataGroupEnd`
      -  `datalogDataGroupStart`
- >  EV3 LED

# 데이터 로깅 - 데이터 로깅 함수 응용 II

## datalogAddValue

**void datalogAddValue(int nDataSeries, int nDataValue)**

Parameter	Explanation	Data Type
Return Type	The function returns no value	void
nDataSeries	Which datalog series to add a value to	int
nDataValue	The value of the data to be added to the datalog series	int

- This command adds a value to the datalog for a given series.
- nDataSeries is the datalog series to add a value to.
  - nDataSeries values range from 0 to 7
- nDataValue is the data to be added to the datalog series
  - The value will not have a time stamp associated with it and will occupy one row in the datalog

# 데이터 로깅 - 데이터 로깅 함수 응용 II

## datalogAddValueWithTimeStamp

**void datalogAddValueWithTimeStamp (nDataSeries, nDataValue)**

Parameter	Explanation	Data Type
Return Type	The function returns no value.	void
nDataSeries	Which datalog series to add a value to.	int
nDataValue	The value of the data to be added to the datalog series	int

- This command adds a value to the datalog for a given series.
- nDataSeries is the datalog series to add a value to.
  - nDataSeries values range from 0 to 7
- nDataValue is the data to be added to the datalog series
  - The value will be time stamped and will occupy one row in the datalog

# 데이터 로깅 - 데이터 로깅 함수 응용 II

```

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     datalogClear();
12
13
14     clearTimer(T1);
15
16
17     while(timer1[T1] < 10000)
18     {
19         setMotorSpeed(lm, 30);
20         setMotorSpeed(rm, -30);
21
22         datalogDataGroupStart();
23         // Add the TS value to datalog series 1
24         datalogAddValue(0, getTouchValue(ts));
25         // Add the GS value to datalog series 2
26         datalogAddValue(1, getGyroDegrees(gs));
27         // Add the CS value to datalog series 3
28         datalogAddValue(2, getColorReflected(cs));
29         // Add the US value to datalog series 4
30         datalogAddValue(3, getUSDistance(us));
31         datalogDataGroupEnd();
32
33         //sleep(200);
34     }
35
36     setMotorSpeed(lm, 0);
37     setMotorSpeed(rm, 0);
38 }
```



# Report

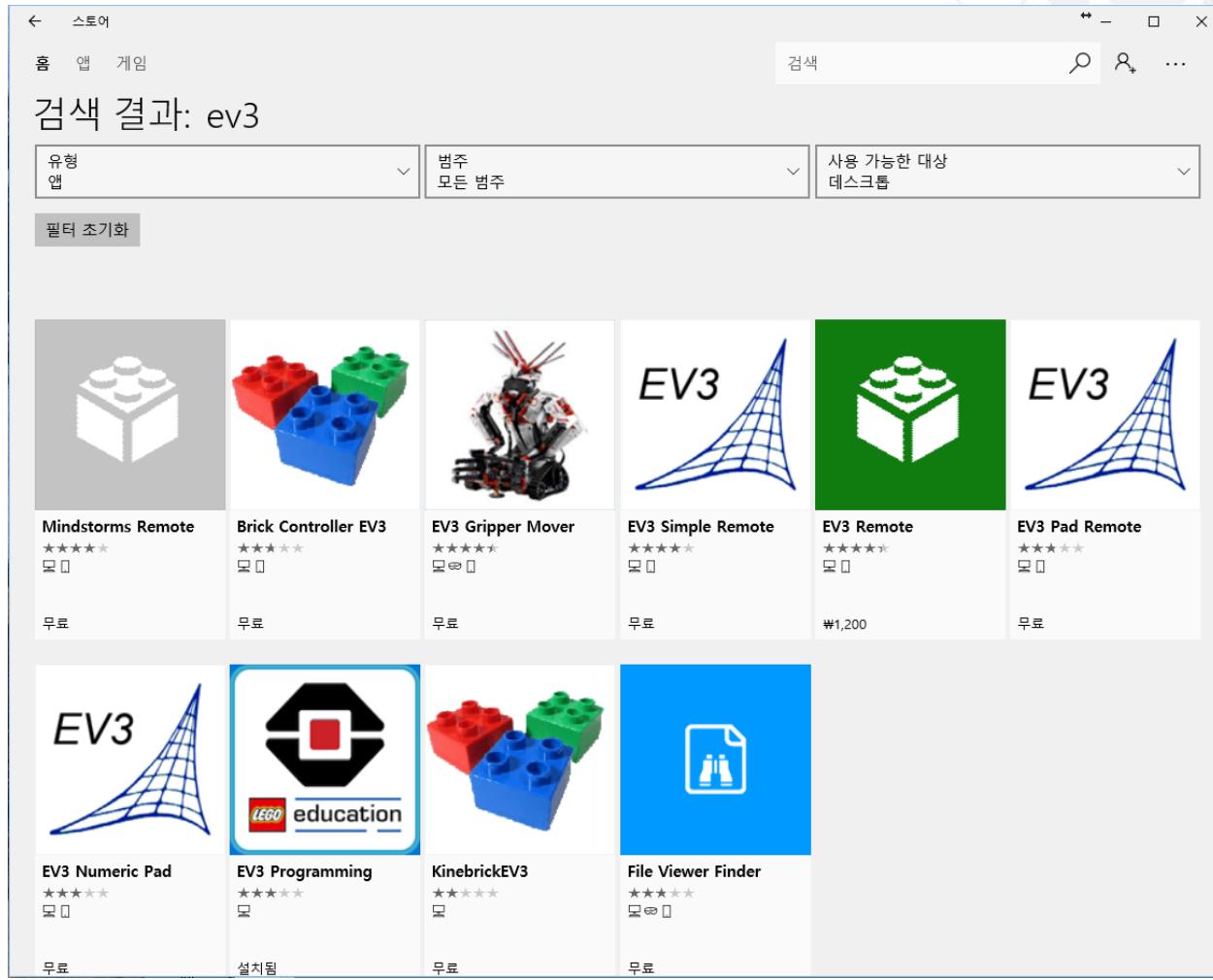
제출파일명 : wk14\_EVnn.zip

- 압축할 파일들

- ① EVnn\_RC\_DL.png
- ② EVnn\_RC\_DL.csv
- ③ EVnn\_RC\_DL2.xls

Email : [chaos21c@gmail.com](mailto: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: ★★★★☆
- Status: 이 제품은 설치되어 있습니다.
- Buttons: 실행 (Run) and 공유 (Share).
- Age Rating: IARC 3세 이상 3+

**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**: Shows three screenshots of the app's user interface, including a robot setup screen, a programming interface, and a control panel.

**모두 표시**



education

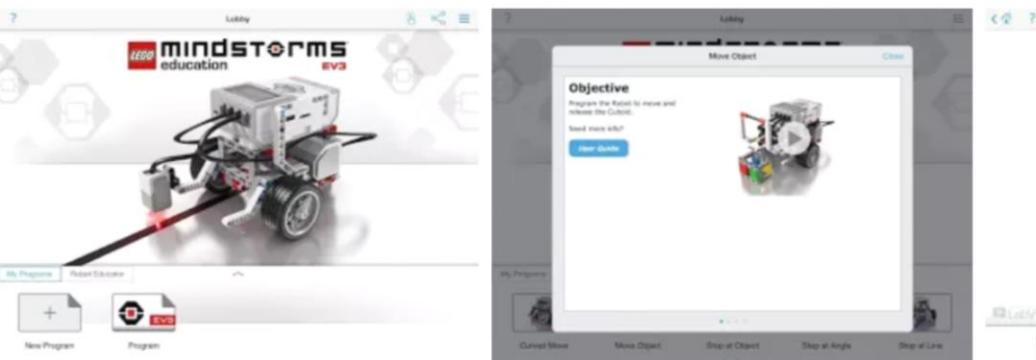
창의공학교육의 멘토

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 app icon features a black and red geometric logo with the word "education" below it. The title is "LEGO® MINDSTORMS® Education EV3" followed by a "4+" rating. It's developed by "LEGO Education". Below the title is a 5-star rating with "(6)" reviews. A blue "OPEN" button is visible. Below the app details are three tabs: "Details", "Reviews", and "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



education

나눔바른고딕

HandsOn  
Technology

# 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