



mBlock 확장기능 만들기 =  + 





# 확장 기능의 동작 방식

확장 기능은 **mBlock**에 대한 사용자 정의 블록을 지원합니다.  
확장 기능을 사용하여 타사의 센서 또는 **LEGO** 또는 **LittleBits**등과 같은 제품을 지원할 수 있습니다.

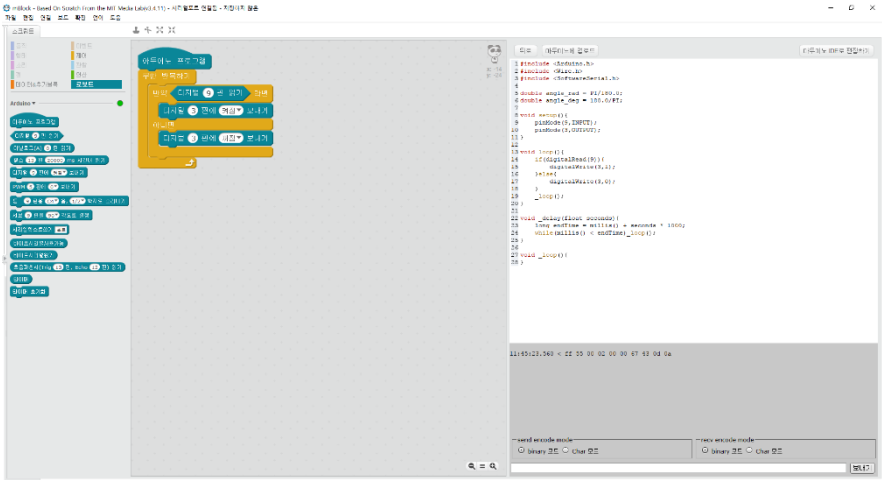
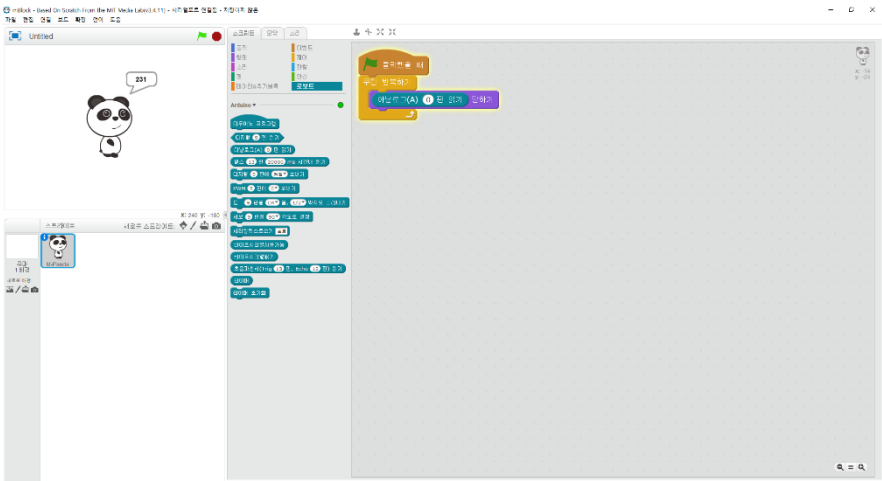
그리고 한번이라도 아두이노 **Sketch**를 사용해보신 분이라면 쉽게 확장기능을 만드실 수 있습니다

## 스크레치 모드와 아두이노 모드

**mBlock**은 기본적으로 두가지 모드로 동작합니다.

-**스크레치 모드**: 스크레치 블록과 확장 블록을 함께 사용하여 스크레치의 스프라이트와 무대를 이용할 수 있습니다.

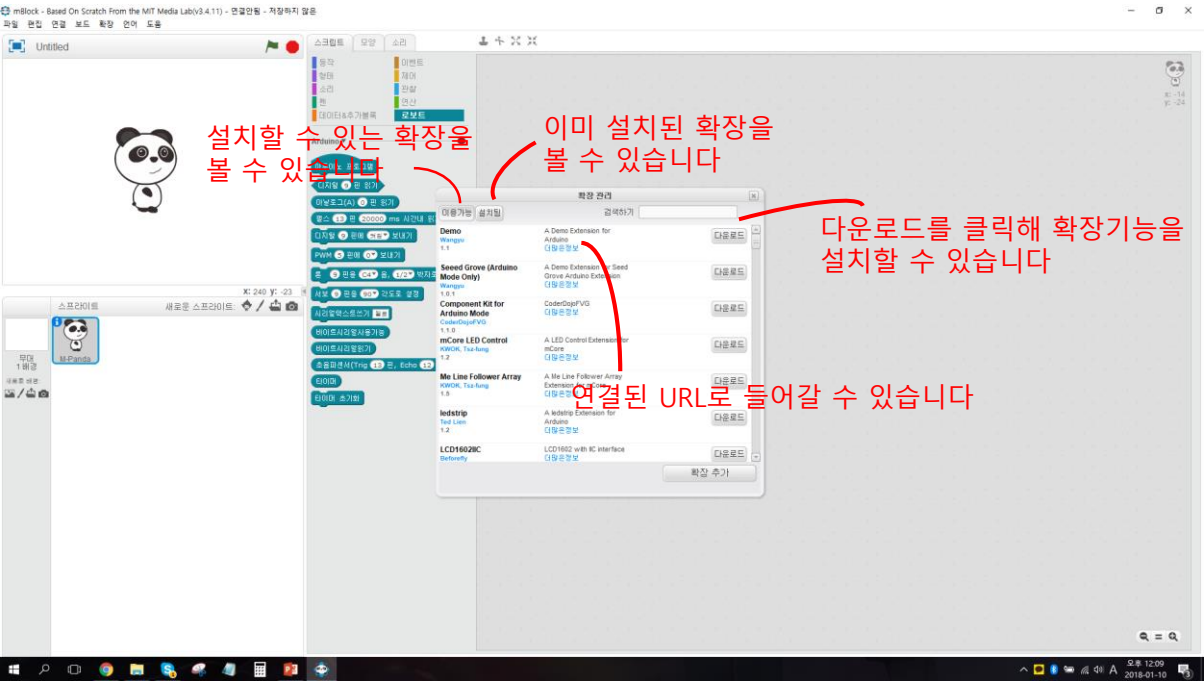
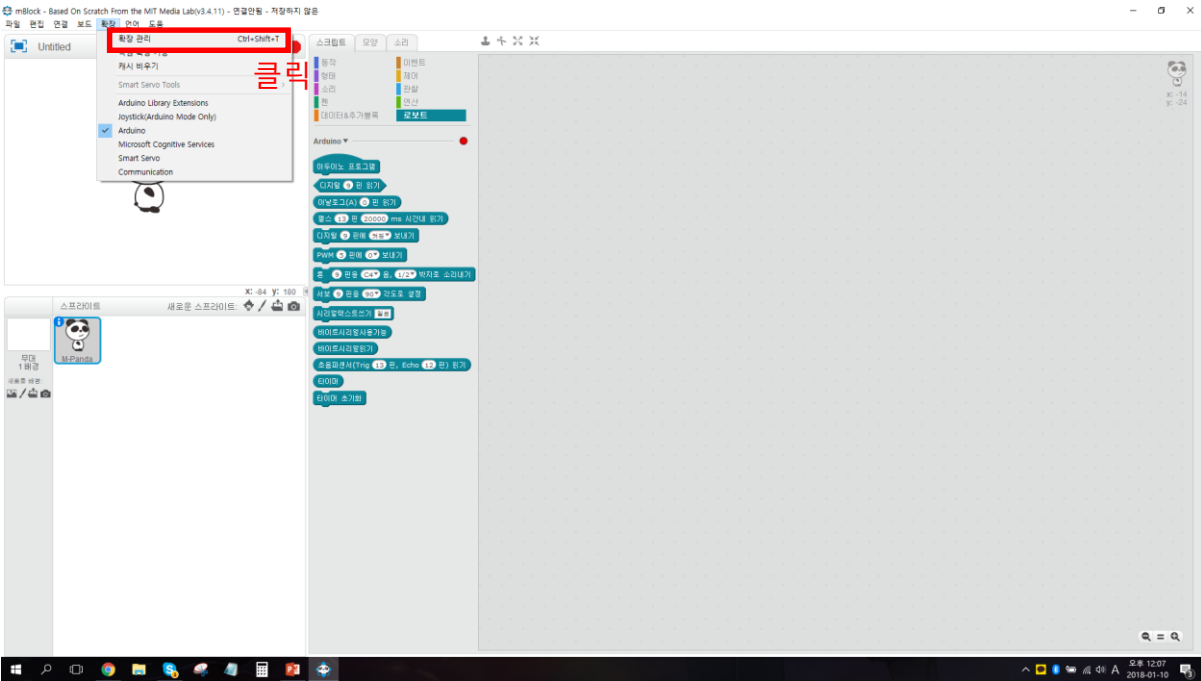
-**아두이노 모드**: 프로그램을 로봇에 업로드할 수 있고 컴퓨터 연결없이 자체적으로 실행 시킬 수 있습니다. 단, 컴퓨터의 연결을 사용하지 않기 때문에 스크레치의 스프라이트와 무대를 사용할 수 없습니다.





# 확장 기능의 사용방법

프로그램 상단메뉴에 “확장”을 클릭하고 “확장관리” 클릭하면 다른 유저가 확장센터에 업로드한 확장기능을 사용할 수 있습니다. 확장기능을 설치하면 “로봇 블록메뉴”에 표시됩니다.





# 확장 기능의 제작하기

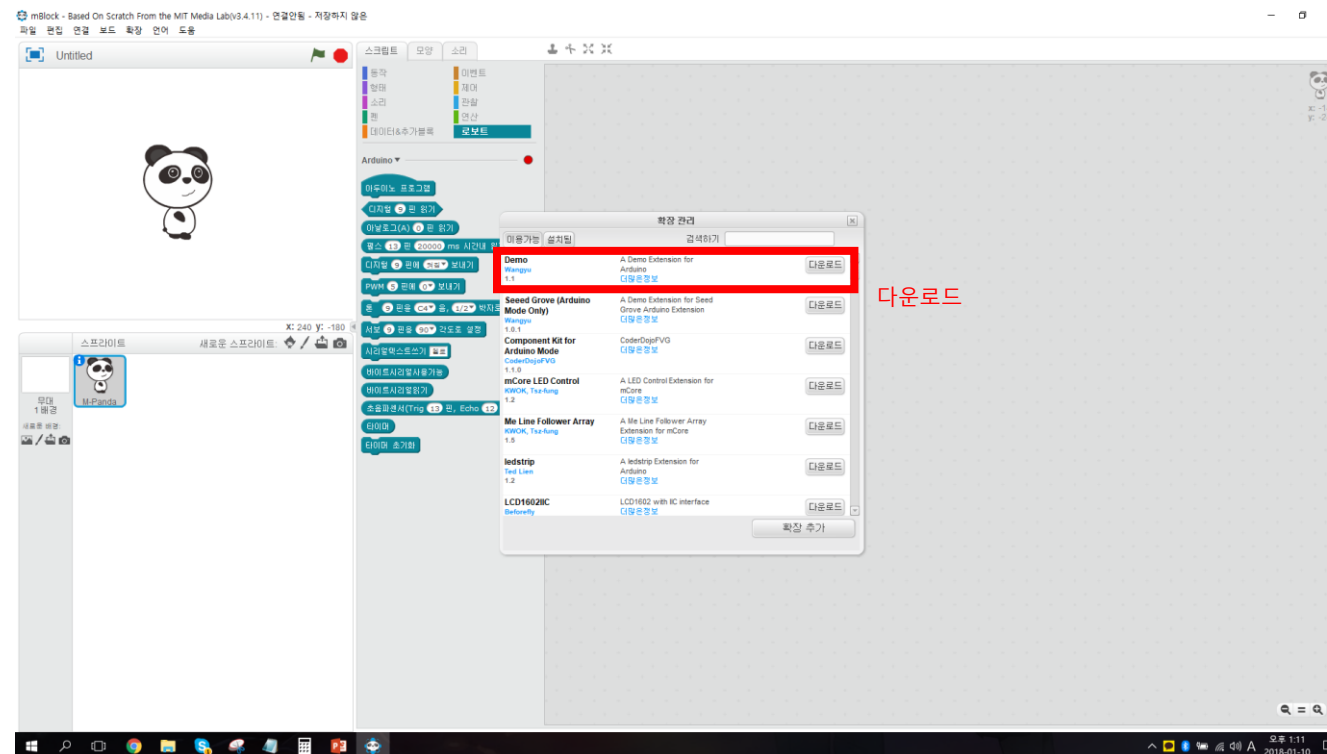
\*참고: 이 문서는 아두이노 확장기능을 만드는 내용만 다루고 있습니다.  
그리고 확장을 제작하기 위해서 기본적으로 C++(아두이노 스케치) 문법을 있어야 합니다.

확장기능은 **JSON, JavaScript, C++(Sketch)**로 제작됩니다. 사용하는 언어가 의외로 많지만 그렇게 어렵지 않습니다. **C++**만 알아도 제작할 수 있습니다. **mBlock**의 스크레치 모드는 **JavaScript**로 구현되고 아두이노는 **JSON**과 **C++**로 구현됩니다. 그런데 **JavaScript**를 사용할 줄 모른다면 스크레치 모드는 굳이 만들 필요 없고 아두이노 모드만 만들어도 됩니다.(물론 스크레치 모드를 건너뛰게 되면 블록이 스크레치 모드에서 동작하지 않습니다.)

다음은 확장 기능 제작 시 해야 될 것들 입니다.

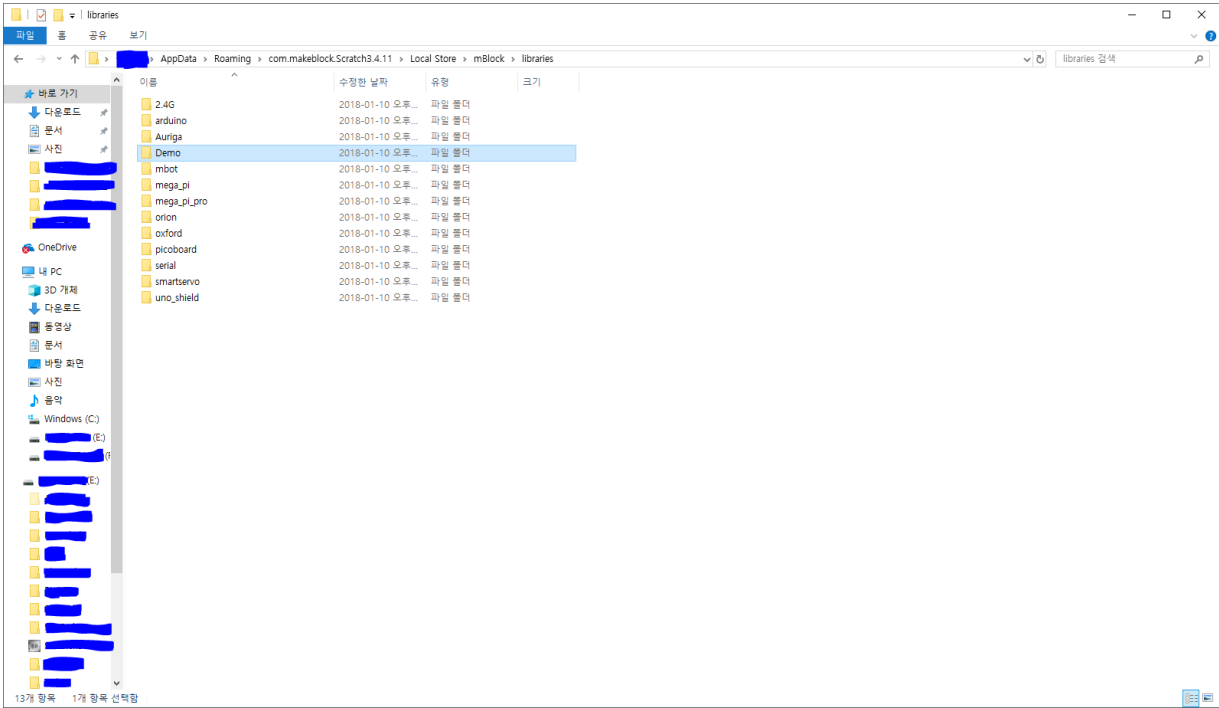
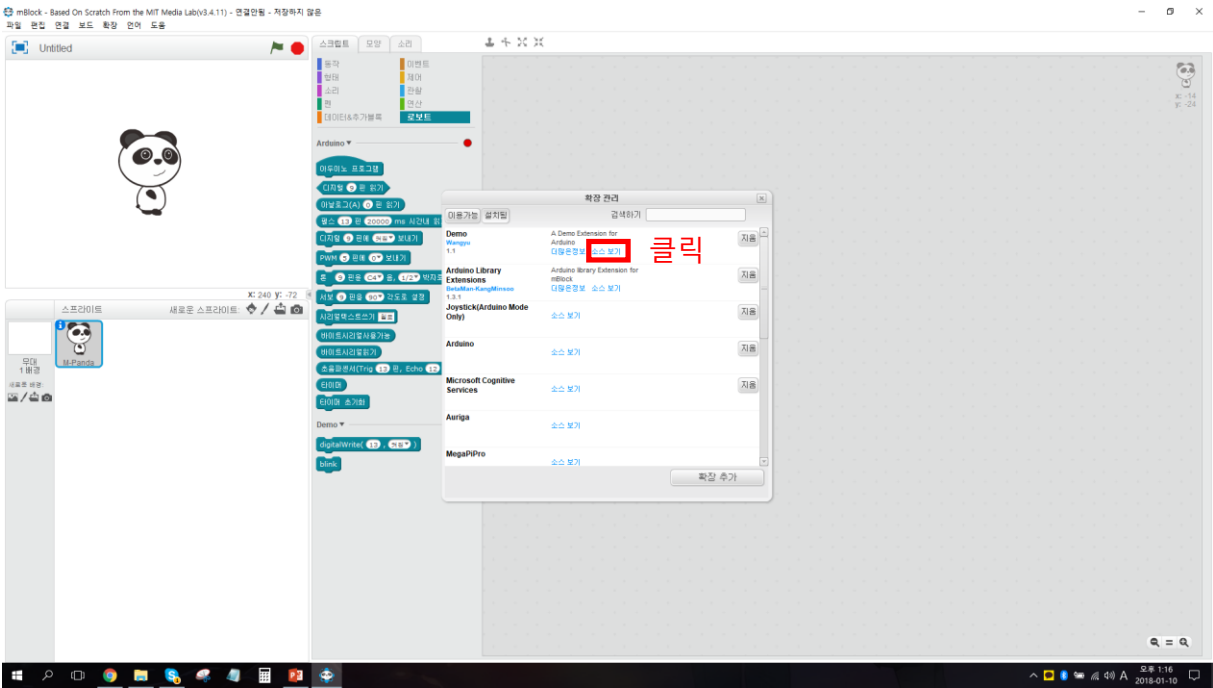
- 확장이름과 저자와 같은 기본 정보를 작성.
- 블록 모양 정의
- **Arduino** 코드를 생성하는 방법 정의하기
- 스크레치 모드에서 실행되는 함수 작성(옵션)
- 추가 라이브러리를 포함하기

간편한 작업을 위해 **mBlock**확장관리에서 **Demo**확장을 다운받아 그것을 수정해 제작할 것입니다.



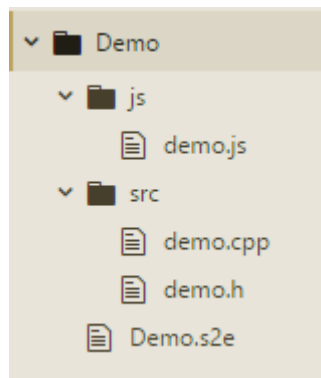


**Demo** 다운받은 뒤 “설치됨” 메뉴에 들어가 **Demo**의 “소스 보기”를 클릭합니다.  
그리고 한 수준 위 폴더로 이동해 **Demo** 폴더를 복사해 작업할 다른 디스크에 저장합니다.





## 확장 기능의 파일 구조



확장 기능은 기본적으로 **js**, **src** 폴더와 **xxx.s2e**파일로 이루어 집니다.

**js** 폴더 안에 **xxx.js**는 스크래치 모드를 구현하는 **JavaScript** 문서이고,  
**src** 폴더 안에 **xxx.cpp**, **xxx.h**는 추가 라이브러리(**C++**) 입니다. 그리고

**xxx.s2e**는 기본적인 확장 정보, 블록의 기능 및 모양 등 담고있는 **JSON** 형식 문서 입니다.

기본 정보

블록 정보

### xxx.s2e 문서의 기본 구조

```

{
  "extensionName": "Demo",
  "description": "A Demo Extension for Arduino",
  "version": "1.1",
  "author": "testAuthor(email)",
  "homepage": "http://www.mblock.cc/posts/create-extensions-for-mblock",
  "sort": 0,
  "javascriptURL": "js/demo.js",
  "firmware": "1.0",
  "extensionPort": 0,
  "tags": "makeblock,demo",
  "blockSpecs": [
    [
      "w",
      "digitalWrite( %n , %d.digital )",
      "digitalWrite",
      "13",
      "HIGH",
      {
        "setup": "pinMode({0},OUTPUT); \n",
        "inc": "",
        "def": "",
        "work": "digitalWrite({0},{1});\n",
        "loop": ""
      }
    ],
    [
      "w",
      "blink",
      "blink",
      {
        "setup": "",
        "inc": "#include \"demo.h\"",
        "def": "DemoClass demo; \n",
        "work": "demo.blink(); \n",
        "loop": ""
      }
    ]
  ],
  "menus": {
    "digital": ["HIGH", "LOW"]
  },
  "values": {
    "HIGH": 1,
    "LOW": 0
  },
  "translators": {
    "zh_CN": {
      "Demo Program": "演示程序",
      "HIGH": "高电平",
      "LOW": "低电平",
      "digitalWrite( %n , %d.digital )": "数字口输出( %n ,%d.digital )",
      "blink": "闪烁"
    }
  }
}

```

메뉴

값 설정

번역

**기본정보:** 이 확장의 이름, 설명, 버전, 작성자, 홈페이지, 표시 순번, **jsURL**, 펌웨어 버전, 태그등의 정보 정의

**블록 정보:** 블록의 모양, 설명, 기능, 코드생성정의 를 정의

**메뉴:** 메뉴 정의

**값 설정:** 메뉴의 필드 값 정의(블록에서 보이는 건 메뉴의 필드, 아두이노 모드에서 보이는 건 필드값)

**번역:** 블록의 언어 번역 정의(굳이 수정할 필요없다)



## 확장 기능의 기본 정보 작성

\*참고: 확장기능을 작성하기 위해서 적절한 에디터를 준비해 주세요. (Atom, brackets, Visual Studio Code, 메모장, 워드패드, notepad++ 등등)

```
{
  "extensionName": "Demo", 확장 이름
  "description": "A Demo Extension for Arduino", 확장 설명
  "version": "1.1", 확장 버전
  "author": "testAuthor(email)", 확장 작성자(이메일)
  "homepage": "http://www.mblock.cc/posts/create-extensions-for-mblock", 홈페이지
  "sort": 0, 확장의 표시 순번
  "javascriptURL": "js/demo.js", Js 문서의 위치
  "firmware": "1.0",
  "extensionPort": 0,
  "tags" : "makeblock,demo",
}
```

기본 확장정보는 밑줄 친 부분만 수정하면 됩니다.(이메일은 옵션)

“sort”는 여러 확장이 표시될 때 이 확장의 표시 순번입니다.(그냥 0으로 하면 됩니다)

Js 문서 경로는 js/문서이름.js를 넣어 주면 됩니다.

버전은 x.y.z 형식으로 작성해주면 됩니다. (y와 z는 생략 가능)

“firmware”, “extensionPort”는 건들면 안되고 “tags”는 삭제를 해도 됩니다.



# 확장 기능의 블록 정의하기

모든 블록은 **JavaScript** 배열 형태로 작성됩니다.

## 블록 모양: 블록의 모양을 정의

모양 옵션:

**h** - 헤더블록을 의미(거의 사용되지 않는다)



**w** - 쓰기블록을 의미(하드웨어에 명령을 보내고 응답을 기대하지 않는다)



**r** - 읽기블록을 의미(값을 바로 읽어올 수 있고, 명령을 보낸 후 응답을 기다릴 수 있다)



**b** - 논리블록을 의미(0 아니면 1을 반환)



## 블록에 넣을 글, 인자: 블록에 표시되는 글과 인자 정의

인자 옵션:

**%n** - 둥근 슬롯을 제공하고 숫자를 받습니다.



**%d.**"이름" - 둥근 드롭 다운 상자를 제공하고 숫자를 받습니다.  
드롭 다운의 내용은 **"menus"**에서 정의되고 **"이름"**으로 식별됩니다.

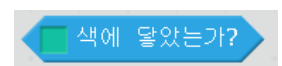
**%s** - 사각형 슬롯을 제공하고 문자열을 받습니다.



**%m.**"이름" - 사각형 드롭 다운 상자를 제공하고 문자열을 받습니다.  
마찬가지로 드롭 다운의 내용은 **"menus"**에서 정의 되고 **"이름"**으로 식별됩니다.



**%c** - 색상 선택기를 제공합니다.



```
"blockSpecs": [
  [
    "w", 블록 모양
    "digitalWrite( %n , %d.digital )", 블록에 넣을 글, 인자
    "digitalWrite",
    "13", 첫번째 인자에 넣을 기본값
    "HIGH", 두번째 인자에 넣을 기본값
    {
      "setup": "pinMode({0},OUTPUT); \n",
      "inc": "",
      "def": "",
      "work": "digitalWrite({0},{1});\n",
      "loop": ""
    }
  ],
  [
    "w",
    "blink",
    "blink",
    {
      "setup": "",
      "inc": "#include \"demo.h\"",
      "def": "DemoClass demo; \n",
      "work": "demo.blink(); \n",
      "loop": ""
    }
  ]
],
```





# 확장 기능의 블록 메뉴 정의하기

```

"blockSpecs": [
  [
    "w",
    "digitalWrite( %n , %d.digital )",
    "digitalWrite",
    "13",
    "HIGH",
    {
      "setup": "pinMode({0},OUTPUT); \n",
      "inc": "",
      "def": "",
      "work": "digitalWrite({0},{1});\n",
      "loop": ""
    }
  ],
  [
    "w",
    "blink",
    "blink",
    {
      "setup": "",
      "inc": "#include \"demo.h\"",
      "def": "DemoClass demo; \n",
      "work": "demo.blink(); \n",
      "loop": ""
    }
  ]
],
"menus": {
  "digital": ["HIGH", "LOW"]
},
"values": {
  "HIGH": 1,
  "LOW": 0
},

```

드롭 다운 상자를 제공하는 인자를 사용하려면 메뉴를 만들어야 합니다.  
메뉴는 다음과 같이 정의됩니다.

"% d.digital"은 메뉴 정보가 메뉴 섹션의 "digital"목록에 저장됨을 의미합니다.

## 자바스크립트 함수 이름 지정하기

```

"blockSpecs": [
  [
    "w",
    "digitalWrite( %n , %d.digital )",
    "digitalWrite",
    "13",
    "HIGH",
    {
      "setup": "pinMode({0},OUTPUT); \n",
      "inc": "",
      "def": "",
      "work": "digitalWrite({0},{1});\n",
      "loop": ""
    }
  ],

```

스크래치 모드에서 사용할 자바 스크립트 함수 이름 지정 ----->

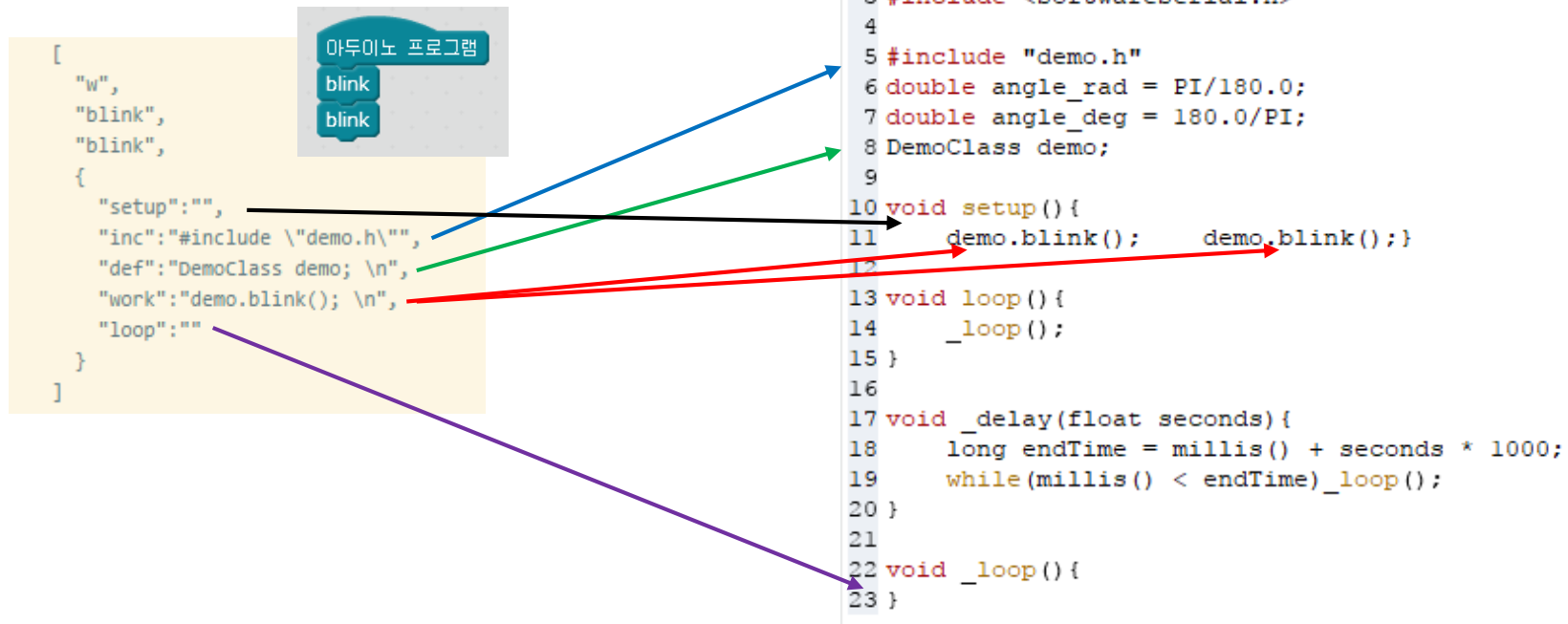
(스크래치 모드에서 확장을 동작시키는 방법이 공개되지 않아  
이 문서에서는 JavaScript 부분을 다루지 않는다. mBlock 공식  
매뉴얼에도 설명이 부족하고 그마저도 제대로 동작하지 않는다.)

Digital 메뉴에 두 가지 옵션이 있음을 의미합니다("HIGH"및 "LOW")

Arduino 모드에서 "HIGH"는 1의 값을 생성하고 "LOW"는 0을 생성합니다.  
하지만 스크래치 모드에는 영향을 미치지 않습니다.



## 아두이노 코드 생성하기



**"setup"**: 블록을 여러 번 사용해도 setup 함수에 딱 "1번" 만 들어갑니다.

**"inc"**: include 문을 포함합니다.(한 번만 들어감)

**"def"**: 변수, 인스턴스, 함수 등을 선언하고 정의합니다.(한 번만 들어감)

**"work"**: 블록을 여러 번 사용하면 사용한 만큼 들어갑니다.(여러 번 들어갈 수 있습니다)

**"loop"**: loop 함수에 딱 "1번" 만 들어갑니다.



아두이노 프로그램

LCD: LCD 1 번 생성, 주소: 0x27

```
[ //lcd I2C interface library
  "w",
  "LCD: LCD %n 번 생성, 주소: %d.addr",
  "LCD begin",
  "1",
  "0x27",
  {
    "setup": "lcd_{0}.init();\nlcd_{0}.backlight();\n",
    "inc": "#include \"LiquidCrystal_I2C.h\"\n",
    "def": "LiquidCrystal_I2C lcd_{0}({1},16,2);\n",
    "work": "",
    "loop": ""
  }
],
```

```
1 #include <Arduino.h>
2 #include <Wire.h>
3 #include <SoftwareSerial.h>
4
5 #include "LiquidCrystal_I2C.h"
6
7 double angle_rad = PI/180.0;
8 double angle_deg = 180.0/PI;
9 LiquidCrystal_I2C lcd_1(0x27,16,2);
10
11 void setup() {
12   lcd_1.init();
13   lcd_1.backlight();
14 }
15
16 void loop() {
17   _loop();
18 }
19
20 void _delay(float seconds){
21   long endTime = millis() + seconds * 1000;
22   while(millis() < endTime) _loop();
23 }
24
25 void _loop() {
26 }
```

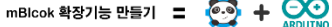
인자를 받아올 때는 {“순번”} 을 적어주면 됩니다.  
따라서 {0} = 1, {1} = 0x27 입니다.

\*여러 문장을 넣어야 될 경우 **NL문자(\n)**를 넣은뒤 다음문장을 작성하면 됩니다.

\*아두이노 코드의 큰 따옴표는 이스케이프 문자(\\)로 입력해야합니다.

\*“inc”에 라이브러리를 추가했으면 **src**폴더에 라이브러리를 넣어야합니다.

\***Arduino.h, Wire.h, SoftwareSerial.h**는 기본으로 포함되어있어 “inc”에 추가하지 않아도 사용이 가능합니다.



확장 기능 제작 팁!: 빨간색으로 표시된 부분처럼 블록 중간에 “-”을  
 넣으면 블록이 한 칸 띄어진다. (“-”의 개수만큼  
 띄어진다. “--” = 두칸)





## .js 파일 수정하기

아직 스크래치 모드에서 작동할 수 있게 .js파일을 수정하는 방법이 공개되지 않아 일단 .s2e파일과 연결하는 방법만 설명합니다.

demo.js

맨 위

```
// demo.js
(function(ext) {
  var device = null;

  var levels = {
    HIGH:1,
    LOW:0
  };

  ext.resetAll = function(){};

  ext.runArduino = function(){

  };
  ext.digitalWrite = function(pin,level) {
    device.send([pin, levels[level]])
  };
  var _level = 0;
  ext.blink = function(){
    device.send([0x22, 0x23])
  }

  function processData(bytes) {
```

같아야 함

Demo.s2e

```
{
  "extensionName": "Demo",
  "description": "A Demo Extension for Arduino",
  "version": "1.1",
  "author": "testAuthor(email)",
  "homepage": "http://www.mblock.cc/posts/create-extensions-for-mblock",
  "sort":0,
  "javascriptURL":"js/demo.js",
  "firmware":"1.0",
  "extensionPort":0,
  "tags" : "makeblock,demo",
```

맨 밑

```
};

ext._getStatus = function() {
  if(!device) return {status: 1, msg: 'demo disconnected'};
  return {status: 2, msg: 'demo connected'};
}

var descriptor = {};
ScratchExtensions.register(demo, descriptor, ext, {type: 'serial'});
})();
```

확장이름으로 수정  
\*꼭 수정할 필요는 없음

\*.js 파일을 수정하지 않기 때문에 스크래치모드에서 동작하지 않지만 아두이노 모드에선 문제없이 잘 동작한다  
 \*추가20180227: 사실 mBlock에서 기본으로 제공하는 아두이노 확장을 보고 만들 수는 있다. 하지만 이 문서에서 js 수정을 다루지 않는 이유를 다음 페이지에서 설명하겠다.



## .js 파일 수정을 다루지 않는 이유

기본적으로 **mBlock**과 아두이노는 정해진 프로토콜을 통해 통신이 이루어진다. (**mBlock**의 **orion** 펌웨어 참고)  
 그래서 예를 들면 **37**번을 요청하면 아두이노에선 펌웨어에 스위치 문으로 정의된 **37**번 케이스 안에 들어있는 코드로 연산 후 그 값을 리턴하거나 출력하는 것이다. 그래서 아두이노 펌웨어가 사용자가 사용할 센서를 지원하지 않는다면 그 센서는 원래 스크레치 모드에서 사용이 불가하다.

하지만 방법은 있다. 그것은 아두이노 펌웨어를 수정하면 된다. 펌웨어 내의 스위치문을 조금만 수정하면 충분히 사용자가 사용할 센서도 스크레치 모드에서 사용이 가능하다. 하지만 이 문서는 **s2e**파일 수정을 중심으로 설명하는 문서이기 때문에 약간 논제에 벗어 나게 된다. 이 펌웨어 수정부분은 나중에 따로 문서화 시켜서 공개할 예정이다.

orion\_firmware

```

84 #define VERSION 0
85 #define ULTRASONIC_SENSOR 1
86 #define TEMPERATURE_SENSOR 2
87 #define LIGHT_SENSOR 3
88 #define POTENTIOMETER 4
89 #define JOYSTICK 5
90 #define BYRD 6
91 #define SOUND_SENSOR 7
92 #define ROBELED 8
93 #define SEVSEG 9
94 #define MOTOR 10
95 #define SERVO 11
96 #define ENCODER 12
97 #define IR 13
98 #define IREMOTE 14
99 #define PIRMOTION 15
100 #define INFRARED 16
101 #define LINEFOLLOWER 17
102 #define IREMOTECODE 18
103 #define SHUTTER 20
104 #define LIMITSWITCH 21
105 #define BUTTON 22
106 #define HUMITURE 23
107 #define FLAMESENSOR 24
108 #define GASSENSOR 25
109 #define COMPASS 26
110 #define DIGITAL 30
111 #define ANALOG 31
112 #define PWM 32
113 #define SERVO_PIN 33
114 #define TONE 34
115 #define PULSEIN 37
116 #define ULTRASONIC_ARDUINO 36
117 #define STEPPER 40
118 #define LEOMATRIX 41
119 #define TIMER 50
120 #define TOUCH_SENSOR 51
121
122 #define GET 1
123 #define RUN 2
124 #define RESET 4
125 #define START 5
  
```

< 프로토콜 리스트

## 사용할 프로토콜 > (Arduino.js)

```

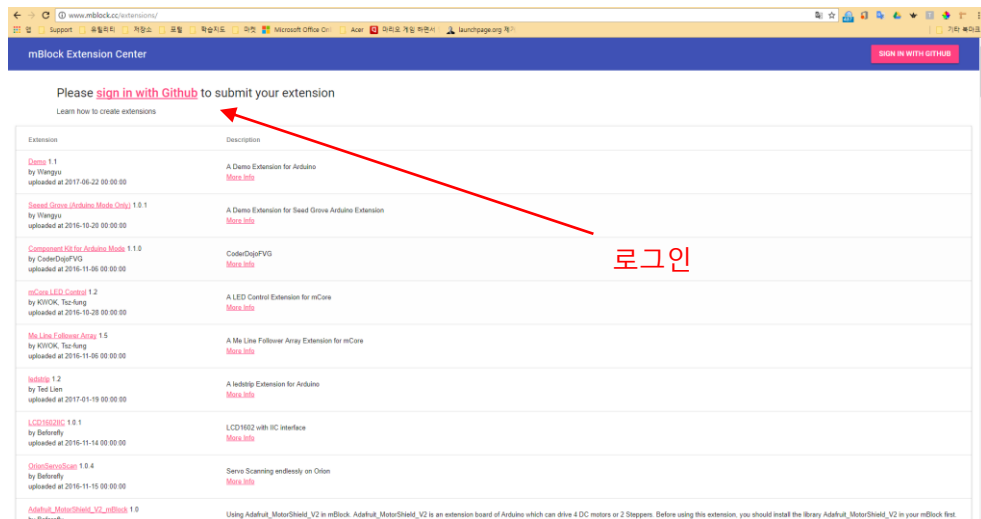
ext.getDigital = function(nextID,pin){
  var deviceId = 30;
  getPackage(nextID,deviceId,pin);
};
ext.getAnalog = function(nextID,pin) {
  var deviceId = 31;
  getPackage(nextID,deviceId,pin);
};
ext.getPulse = function(nextID,pin,timeout) {
  var deviceId = 37;
  getPackage(nextID,deviceId,pin,short2array(timeout));
};
ext.getUltrasonicArduino = function(nextID,trig,echo){
  var deviceId = 36;
  getPackage(nextID,deviceId,trig,echo);
}
ext.getTimer = function(nextID){
  if(startTimer==0){
    startTimer = new Date().getTime();
  }
  responseValue(nextID,new Date().getTime()-startTimer);
}
  
```

## <정의된 프로토콜

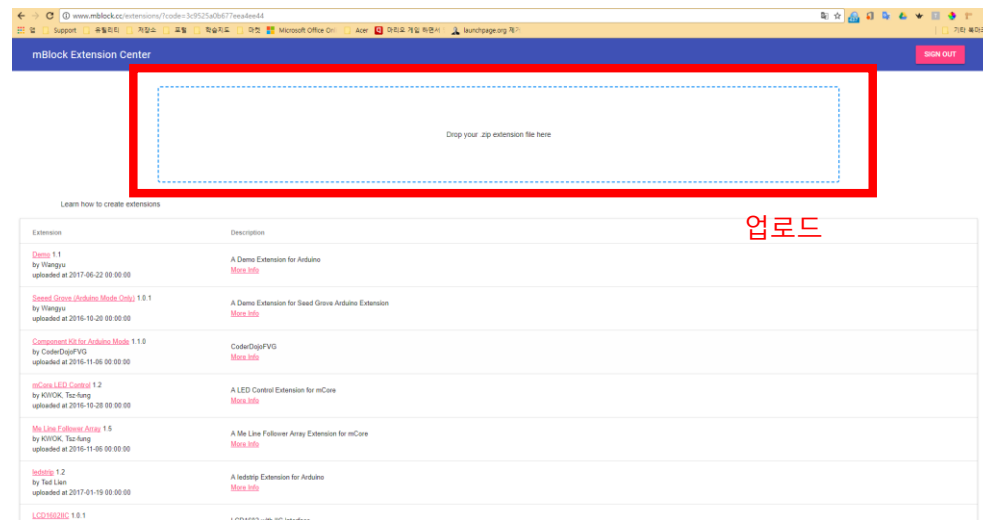


# 확장 기능 공유하기

다 완성이 되면 폴더 채 압축하고 <http://www.mblock.cc/extensions/> 여기에 들어간 뒤 **GitHub** 계정으로 로그인을 한다.  
(GitHub 계정이 없다면 만들어야 합니다.)



.zip 파일을 드래그하거나 파란 테두리 박스를 클릭해 업로드 한다.  
(\*참고로 한번 올려버린 확장은 삭제와 다운그레이드가 불가능합니다.)



\*확장을 업로드 하기 전에 테스트를 할 때는 확장관리에 확장추가를 이용해 .zip파일을 설치하면 됩니다.

\*확장 업데이트를 하기 위해선 .s2e파일의 “version”이 업로드 된 확장보다 보다 더 높아야 한다.




# 확장 기능 제작 팁

- 아무 이유없이 컴파일&업로드 에러가 난다면?
  - > **mBlock**을 재시작 해보고 안된다면 아래 설명을 따라 진행합니다.
  - 1. 일단 프로젝트를 저장합니다.
  - 2. **mBlock**을 종료하고 파일탐색기에서 아래 경로로 이동합니다.  
(C:\Users\사용자이름\AppData\Roaming\com.makeblock.Scratch3.4.11\Local Store\scratchTemp)
  - 3. 그 안에 있는 모든 폴더를 삭제합니다.
  - 4. 이제 **mBlock**을 켜고 다시한번 컴파일&업로드를 해봅니다.  
(만약 이래도 안된다면 확장기능이나 포함된 라이브러리에 예외가 발생한 것입니다.)
- 디버깅은 어떻게 해야 하는가?
  - > **s2e** 파일은 블록의 설명과 그 안에 들어가는 스케치를 정의하는 **json** 파일입니다. 그래서 일반적으로 웹 **json** 검사기를 이용해 **s2e** 파일의 유효성을 검사하고,  
**s2e** 파일에 스케치를 넣을 때 그냥 **s2e** 파일 내에서 즉석으로 작성하지 말고 미리 아두이노 스케치 에디터에서 먼저 작성, 디버깅한 뒤 **s2e** 파일에 붙여 넣는 것을 추천드립니다.
- 공유한 확장 기능에 버그가 발생했다면?
  - > 확장센터에서 공유한 확장 기능을 삭제하고 버그를 수정한 확장 기능을 다시 올리거나 새로운 버전으로 올리면 됩니다. (삭제보다는 새로운 버전으로 공유하는 것을 추천드립니다.)





끝까지 보시느라 수고 많으셨습니다. 감사합니다

mBlock 확장기능 만들기 =  + 