



연암대학교

B to Y

아두이노 문법 총정리

연암대학교 2024 스마트팜 프로젝트

INDEX

I. 아두이노 IDE 설치하기

V. 기초 문법

II. 라이브러리 총 정리

VI. 마무리

III. 변수와 데이터 타입

IV. 연산자

I. 아두이노 IDE 설치하기

IDE? = 통합 개발 환경

아두이노 소스 코드를 작성하고 코드를 업로드 할 수 있는 곳

Downloads



Arduino IDE 2.3.3

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits

Windows MSI installer

Windows ZIP file

Linux Appliance 64 bits (X86-64)

Linux ZIP file 64 bits (X86-64)

macOS Intel, 10.15: "Catalina" or newer, 64 bits

macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

알맞은 운영체제를 선택하여

다운로드하면 된다.

(사진에 하이퍼 링크 첨부)

II. 라이브러리 총 정리

라이브러리?

의미 그대로 **도서관 (= 책들의 집합)**.

미리 기능들을 모아둔 **집합소(도서관)**에서 필요할 때마다 기록된 코드를 빼와서 바로 사용 할 수 있도록 만들어 진 것.

라이브러리는 왜 사용하는 것일까??

- (1) 모든 기능을 직접 코딩할 수 없어서(어려워서)
- (2) 업무 효율을 위한 시간 절약을 위해서

II. 라이브러리 총 정리

- 하나하나 코딩-

1. 삼각형1 코딩
2. 삼각형2 코딩
- .
- .
- .
10. 삼각형10 코딩
11. 사각형1 코딩
12. 사각형2 코딩
- .
- .
- .
20. 사각형10 코딩
21. 원1 코딩
22. 원2 코딩
- .
- .
- .
30. 원10 코딩

- 라이브러리 사용-

1. 라이브러리 작성
라이브러리 명: 도형 그리기
책1 - 삼각형 만들기
책2 - 사각형 만들기
책3 - 원 만들기
2. 라이브러리 불러오기
3. 라이브러리 사용하여 도형 찍어내기

← 라이브러리의 기능을
이해하기 쉽게 설명한 예시

II. 라이브러리 총 정리

라이브러리 사용법

```
#include <라이브러리 이름.h>
```

라이브러리를 사용하기 전 라이브러리가 (1)다운로드 되었는지, (2)폴더 내에 위치하여 있는지 확인해야 한다.

(몇몇 라이브러리는 이미 다운로드 되어있을 수 있다 때문에 바로 쓰면 된다.)

IDE 상단 `Sketch > Include Library`를 통해 다운로드한 라이브러리를 확인하고 코드에 추가할 수 있다.

III. 변수와 데이터 타입

int: 정수 (1,2,3,4,5) `int a = 5;`

float: 실수, 소숫점 (3.14152) `float b = 5.67;`

bool, boolean : true, false `bool LedState = true; //boolean도 가능`

char: 단일 문자 (a, b, c) `char c = 'D';`

string: 문자열 (Hello World) `string d= 'Hello World';`

항상 변수의 데이터 타입에 맞는 변수형을 선언하여 사용한다.

IV. 연산자

연산자

수학적 연산에 사용되는 기호

아두이노의 연산자는 C/C++언어와 같다.


LCD등 가시적 구분을 위해 문자를 붙이는 것 외 모든 데이터는 숫자로 이루어져 있기 때문에 자주 쓰이는 연산자는 외우는 것이 좋다.

IV. 연산자

연산자	의미	설명	사용 예	결과 및 해석	비고
=	대입	연산자 오른쪽의 값을 왼쪽 변수에 대입한다.	ex1) led=10; ex2) a=10; led = a;	ex1) 변수 led에 10을 대입한다. ex2) 변수 a에 10을 대입하고, 변수 a를 변수 led에 대입한다. 따라서 led값은 모두 10이 된다.	주의 : 같다(equal)는 ==을 사용한다.
+	더하기	피연산자의 값을 더한다.	ex1) led=10+3; ex2) a=10; led = a+3;	ex1) 10+3을 한 다음 변수 led에 대입한다. ex2) 변수 a에 10을 대입하고, 변수 a와 3을 더하여 변수 led에 대입한다. 따라서 led값은 모두 13이 된다.	
-	빼기	피연산자의 왼쪽 값에서 오른쪽 값을 뺀다.	ex1) led=10-3; ex2) a=10; led = a - 3;	ex1) 10-3을 한 다음 변수 led에 대입한다. ex2) 변수 a에 10을 대입하고, 변수 a에서 3을 뺀 다음 변수 led에 대입한다. 따라서 led값은 모두 7이 된다.	
*	곱하기	피연산자의 값을 곱한다.	a = 10; led = a * 2;	변수 a에 10을 대입하고, 변수 a와 2를 곱하여 변수 led에 대입한다. 따라서 led값은 20이 된다.	
/	나누기	피연산자의 왼쪽 값을 오른쪽 값으로 나눈다.	a = 10; led = a / 2;	변수 a에 10을 대입하고, 변수 a를 2로 나누어 변수 led에 대입한다. 따라서 led값은 5가 된다.	주의1 : 0으로 나눌 수 없다. 주의2 : 정수형으로 나눌 경우 소수점은 버려진다. ex) int led = 3 / 2; 위와 같은 경우 led값은 1이다. 소수점을 얻고자 한다면 실수형 변수를 사용하고 3.0 / 2.0과 같이 소수점을 표시해 주어야 한다.
%	나머지	피연산자의 왼쪽 값을 오른쪽 값으로 나눴을 때 구해지는 나머지를 나타낸다.	a=9; led = a % 2;	변수 a에 9를 대입하고, 변수 a를 2로 나눈 나머지 값을 led에 대입한다. 따라서 led 값은 1이 된다.	

	연산자	사용 예	결과 및 해석	비고
산술 연산	+=	led += a;	led = led + a;	
	-=	led -= a;	led = led - a;	
	*=	led *= a;	led = led * a;	
	/=	led /= a;	led = led / a;	
비트 연산	&=	led &= a;	led = led & a;	
	=	led = a;	led = led a;	
	^=	led ^= a;	led = led ^ a;	
	<<=	led <<= a;	led = led << a;	
증감 연산	>>=	led >>= a;	led = led >> a;	
	++	led = ++a; led = a++;	led = a + 1;	++a의 경우 선 증가 후 연산 a++의 경우 선 연산 후 증가
	--	led = --a; led = a--;	led = a - 1;	--a의 경우 선 감소 후 연산 a--의 경우 선 연산 후 감소



연산자	의미	설명	사용 예 (a=10, b=3이라 가정)	결과 및 해석	비고
>	크다	연산자 왼쪽 값이 오른쪽 값보다 큰가?	led=a>b;	a의 값은 10이고 b의 값은 3이고, 10은 3보다 크므로 참이 된다. 따라서 led값은 1이 된다.	led = b > a;라고 하면 컴파일 에러가 아니라 결과는 거짓이므로 led에 0을 대입하게 된다.
>=	크거나 같다	연산자 왼쪽 값이 오른쪽 값보다 크거나 같은가?	ex1) led=(a>=b); ex2) led=(a>=10); ex3) led=(a>=11);	ex1) a가 b보다 크므로 참 led값은 1 ex2) a와 10은 같으므로 참 led값은 1 ex3) a는 11보다 작으므로 거짓 led값은 0	
<	작다	연산자 왼쪽 값이 오른쪽 값보다 작은가?	led = a < b;	a의 값은 10이고 b의 값은 3이고, 10은 3보다 크므로 거짓이 된다. 따라서 led값은 0이 된다.	
<=	작거나 같다	연산자 왼쪽 값이 오른쪽 값보다 작거나 같은가?	ex1) led=(a<=b); ex2) led=(a<=10); ex3) led=(a<=11);	ex1) a가 b보다 크므로 거짓 led값은 0 ex2) a와 10은 같으므로 참 led값은 1 ex3) a는 11보다 작으므로 참 led값은 1	
==	같다	연산자 왼쪽 값과 오른쪽 값이 같은가?	ex1) led=(a==b); ex2) led=(a==10);	ex1) a와 b는 같지 않으므로 거짓 led값은 0 ex2) a와 10은 같으므로 참 led값은 1	주의 : 대입 연산자와는 등호 하나 =를 사용하고, 등호(equal)는 ==을 사용한다.
!=	다르다	연산자 왼쪽 값과 오른쪽 값이 다른가?	led = (a != b);	a와 b는 같지 않으므로 참 led 값은 1	

<https://codingrun.com/93>

V. 기초 문법

개요:규칙

1. 아두이노는 C/C++의 문법 구조와 유사하며, C/C++와 같은 변수타입과 선언 방식을 가진다.
2. #으로 시작하는 라이브러리, define과 중괄호{}를 제외한 모든 문장의 끝에는 세미콜론(;)을 붙인다.
3. 주석은 ‘//’로 쓰며 여러 줄에 동시에 주석을 붙일 때는 /*(시작)과 */(끝)으로 주석처리 한다.
주석 처리된 문장은 회색이 되며, 비활성화 된다(기능X)

예1) //한 줄 주석

예2) /* 여
러
줄
주
석*/

V. 기초 문법

기본구조:두 가지 주요 함수

아두이노 파일을 생성하면 기본으로 작성되어 있으며, 코드 동작의 필수 구성요소

```
void setup() {  
  //초기화 코드  
}
```

setup(): 보드가 전원이 켜지거나 리셋될 때 한 번 실행. 설정을 초기화하는 데 사용.

```
void loop() {  
  //반복할 코드  
}
```

loop(): setup() 함수가 끝난 후 계속 무한 반복 실행. 반복 작업에 사용.

V. 기초 문법

시리얼 통신

시리얼 통신 초기화

```
Serial.begin(9600);
```

시리얼 통신 속도 설정 보편적으로 9600으로 설정한다.

데이터 전송

```
Serial.print(""); //시리얼 문자 출력
```

```
Serial.println(""); //시리얼 문자 출력 + 단 바꿈(Enter)
```

시리얼 모니터에 출력, 데이터 전송

시리얼 수신

```
Serial.read();
```

시리얼 값을 읽어온다. 잘 안 씀

V. 기초 문법

입출력

핀 모드 설정

```
pinMode(13, OUTPUT); //13번 핀을 출력 모드로 설정하겠다.
```

13번 핀은 출력 상태가 됨, 만약 센서를 이용할 때는 센서로 부터 입력 받아야 하므로 INPUT을 써야한다.

디지털 제어

```
digitalWrite(13, HIGH); //13번 핀에 5V 출력
```

```
int buttonState = digitalRead(2); //2번 핀의 입력 상태 읽기
```

디지털 신호를 제어하거나 읽어온다. 장치 제어 시 digitalWrite를 자주 쓴다.

아날로그 제어

```
analogWrite(9, 128); //9번 핀에 PWM 신호 출력
```

```
int sensorValue = analogRead(A0); //A0핀의 아날로그 입력 읽기
```

아날로그 형식의 센서와 장치를 쓸 때 이용한다.

V. 기초 문법

간단한 예제1

```
1. int ledPin = 13; // LED가 연결된 핀 번호
2. void setup() {
3.   pinMode(ledPin, OUTPUT); // LED 핀을 출력으로 설정
4. }
5. void loop() {
6.   digitalWrite(ledPin, HIGH); // LED 켜기
7.   delay(1000); // 1초 대기 1000ms
8.   digitalWrite(ledPin, LOW); // LED 끄기
9.   delay(1000); // 1초 대기
10. }
```

LED를 1초 간격으로 켜다가 키는 코드

아두이노에서 가장 보편적인 예제로 변수 선언방법, 핀모드 설정, 디지털 신호 출력(digitalWrite)을 한 번에 이해할 수 있다.

13번 핀을 ledPin이라는 이름으로 변수 선언(int)후, setup에서 핀모드를 출력으로 설정, loop부분에서 디지털 신호를 1초 마다 HIGH, LOW로 바꾼다.

V. 기초 문법

간단한 예제2

```
1. int ledPin = 13; // LED 핀이 연결된 핀 번호
2. int buttonPin = 2; // 버튼이 연결된 핀 번호
3. int buttonState = 0; // 버튼 상태 저장 변수
4. void setup() {
5.     pinMode(ledPin, OUTPUT); // LED 핀을 출력 모드로 설정
6.     pinMode(buttonPin, INPUT); // 버튼 핀을 입력 모드로 설정
7. }
8. void loop() {
9.     buttonState = digitalRead(buttonPin); // 버튼의 상태 읽기
10.    if (buttonState == HIGH) {
11.        digitalWrite(ledPin, HIGH); // 버튼이 눌리면 LED 켜기
```

```
12.    } else {
13.        digitalWrite(ledPin, LOW); // 버튼이 눌리지 않으면 LED 끄기
14.    }
15. }
```

버튼으로 LED를 제어하는 코드로 핀모드 INPUT, OUTPUT의 개념과,

if문의 구조를 한 번에 이해할 수 있는 코드

LED, 버튼의 핀 번호를 선언

버튼 상태를 저장하는 변수 0, 1 (초기 상태는 0=LOW)

setup에서 핀모드 설정 LED는 출력(OUTPUT) 버튼은 입력(INPUT)

loop에서 읽고 if문을 이용해 버튼 상태에 따라 LED 온/오프

V. 기초 문법

간단한 예제3

```
1. int sensorPin = A0; // 센서가 연결된 아날로그 핀 번호
2. int sensorValue = 0; // 센서 값 저장 변수
3. void setup() {
4.   Serial.begin(9600); // 시리얼 통신 속도 설정
5. }
6. void loop() {
7.   sensorValue = analogRead(sensorPin); // 센서 값을 읽기
8.   Serial.print("조도 센서 값: ");
9.   Serial.println(sensorValue); // 시리얼 모니터에 값 출력
10.  delay(500); // 0.5초 대기
11. }
```

아날로그 방식의 센서를 이용해 출력값을 시리얼 데이터로 출력하는 코드이다.

아날로그 센서 사용법과 시리얼 데이터 출력을 이해할 수 있다.

Serial을 사용하므로 시리얼 통신 속도를 설정하여야 한다.

analogRead로 시리얼 센서 핀의 아날로그 입력값을 읽은 뒤, print를 통해 한 줄마다 출력한다.

출력값 예시

167

167

168

168

168 ...

V. 기초 문법

delay와 millis함수와 RTC모듈

아두이노로 장치를 제어할 때 장치의 작동시간을 조절해야 하는 일이 생긴다.

이번 프로젝트에서는 낮과 밤시간 마다 LED를 온/오프 해줘야 하고, 모터펌프를 주기적으로 일정 시간 작동시켜 뿌리가 마르지 않게 주기적으로 작동시켜야 한다.

이때 delay나 millis함수 또는 RTC모듈을 사용하게 된다. loop 안의 delay는 다음 코드가 동작할 때 까지 코드 자체를 지연(delay) 시키기 때문에 여러 동작을 동시에 수행해야 하는 코드에서 쓰이기 힘들다.

따라서 millis함수나 RTC모듈을 사용해야 한다.

날짜 데이터를 따로 수집하기 위해 RTC모듈을 장착했기 때문에 이번 프로젝트에서는 RTC모듈을 사용하여 시간을 장치에 이용할 거지만, millis와 RTC모듈은 작동방식이 비슷하고 millis함수 또한 자주 쓰이기 때문에 둘 다 자료에 추가하였다.



←RTC모듈

V. 기초 문법

millis함수

millis함수는 코드가 시작된 이후 경과된 시간을 밀리초 단위로 반환한다.

작동시작 → millis 동작 1,2,3,4,5...

이전millis와 현재millis 두 개의 변수를 이용하여 시간을 비교 한다. 예를 들어 1초 마다 LED를 온/오프하고 싶다면 이전 시간과 현재 시간이1000(ms)차이가 날 때 LED의 상태를 변경하면 된다.

예) 이전 시간 = 7000(ms), 현재 시간 = 8000(ms) → LED상태 변경 →이전 시간을 현재 시간으로 갱신 → 이전 시간 = 8000(ms), 현재 시간 = 9000(ms) → LED상태 변경...(반복)

V. 기초 문법

millis함수

코드로 보면 이러하다

```
1. const int ledPin = 13; // LED 핀 번호
2. unsigned long previousMillis = 0; // 이전 시간 저장 변수
3. const long interval = 1000; // 간격을 1000ms(1초)로 설정
4. void setup() {
5.     pinMode(ledPin, OUTPUT); // LED 핀을 출력 모드로 설정
6. }
7. void loop() {
8.     unsigned long currentMillis = millis(); // 현재 시간을 millis()로 얻음
9.     if (currentMillis - previousMillis >= interval) {
10.         previousMillis = currentMillis; // 이전 시간을 현재 시간으로 갱신
11.         if (digitalRead(ledPin) == LOW) {
12.             digitalWrite(ledPin, HIGH); // LED 켜기
13.         } else {
```

```
14.             digitalWrite(ledPin, LOW); // LED 끄기
15.         }
16.     }
17. }
```

millis함수는 라이브러리를 이용하지 않고도 쓸 수 있을 정도로 코드가 간단해 진다는 장점이 있지만, 32bit의 long 변수로 계산하므로 2의32 제곱 밀리초인 (약 49.71일) 뒤에 오버플로우(overflow)되어 0으로 돌아가 시간 계산에 오류가 생길 수 있어 장시간 구동되는 동시에 정확성과 안정성을 요구하는 이번 프로젝트의 취지와 맞지 않아 millis함수는 쓰지 않을 것이다.

V. 기초 문법

RTC모듈

RTC는 'real time clock'의 약자로 직역하면 '실시간 시계'이다.

의미 그대로 그냥 일반적인 전자식 시계라고 생각하면 된다.

RTC모듈도 millis함수와 같이 이전시간과 현재시간의 차이를 이용해 원하는 시간에 디지털 신호를 변경할 수 있다.

다만 RTC는 아두이노에 연결하여 사용하는 하드웨어 모듈이므로 별도의 라이브러리를 코드에 불러와 이용한다.

다음 슬라이드는 RTC모듈을 이용해 LED상태를 1초 마다 변경하는 코드 예제이다.

V. 기초 문법

```
1. #include <Wire.h>
2. #include <RTCLib.h>
3. RTC_DS3231 rtc; // RTC 객체 생성
4. const int ledPin = 13; // LED 핀 번호
5. DateTime previousTime; // 이전 시간을 저장할 변수
6. const int interval = 1; // 간격을 1초로 설정
7. void setup() {
8.     pinMode(ledPin, OUTPUT); // LED 핀을 출력 모드로 설정
9.     Serial.begin(9600);
10.    // RTC 초기화
11.    if (!rtc.begin()) {
12.        Serial.println("RTC 연결 실패!");
13.        while (1);
14.    }
15.    // RTC 시간이 맞춰지지 않았다면 설정 (처음 사용 시 필요)
16.    if (rtc.lostPower()) {
17.        Serial.println("RTC 시간 설정 중...");
```

```
18.     rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
19. }
20. previousTime = rtc.now(); // 이전 시간을 현재 시간으로 설정
21. }
22. void loop() {
23.     DateTime currentTime = rtc.now(); // 현재 시간을 얻음
24.     // 일정 시간이 지났는지 확인 (1초)
25.     if (currentTime.second() != previousTime.second()) {
26.         previousTime = currentTime; // 이전 시간을 현재 시간으로 갱신
27.         if (digitalRead(ledPin) == LOW) {
28.             digitalWrite(ledPin, HIGH); // LED 켜기
29.         } else {
30.             digitalWrite(ledPin, LOW); // LED 끄기
31.         }
32.     }
33. }
```

VI. 마무리

이 자료는 연암대학교 2024 스마트팜 제작 프로젝트의 아두이노 코드의 이해를 돕기 위해 만들어진 자료입니다.

자료를 숙지한다면 대부분의 코드를 이해하고, 간단한 코드를 쓸 수 있을 정도로 꼭 필요한 내용들만 담았습니다.

따라서 프로젝트에 필요하지 않은 일부 기본적인 이론과 문법이 생략되었고, 복잡하거나 어려운 코드는 제외하여 만들었습니다.

모르거나 이해가 되지 않는 내용은 온라인 상에 다양한 예제가 있으니 이해가 될 때 까지 따로 공부해주세요.