

아두이노 프로그래밍

1일차 - Part4 값출력 및 디지털 읽기

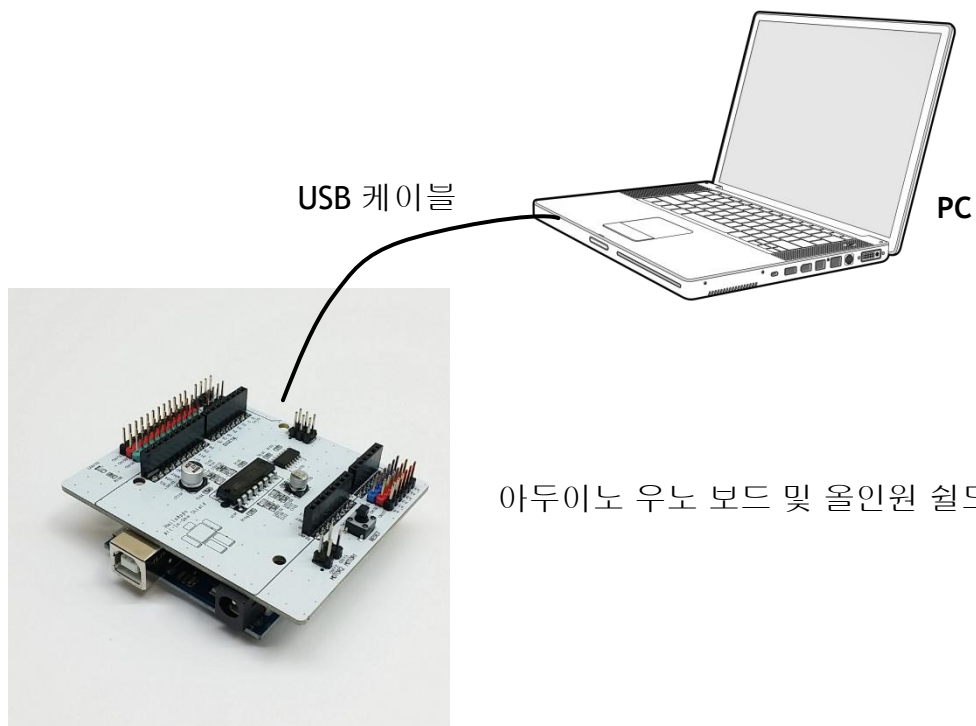
강사: 김영준 헬로앱스 대표
헬로앱스 (www.helloapps.co.kr)

아두이노와 PC의 통신

아두이노와 PC의 통신

아두이노 통신

아두이노 보드와 PC가 USB 케이블을 통해 연결되어 있는 경우



시리얼 통신으로
PC와 아두이노 보드가 데이터를
주고 받을 수 있습니다.

아두이노 -> PC 값 전송하기

아두이노 통신

- ▶ 아두이노 보드에서 PC 또는 외부로 값을 보내기 위해서는 다음의 명령어를 사용합니다.

Print(값)

- 값을 이어서 출력합니다.

PrintLine(값)

- 값을 출력한 후, 줄을 바꿉니다.

실제 스케치 코드는 `Serial.println(값);` 형태임

실습

아두이노 -> PC 값 전송하기

아두이노 통신

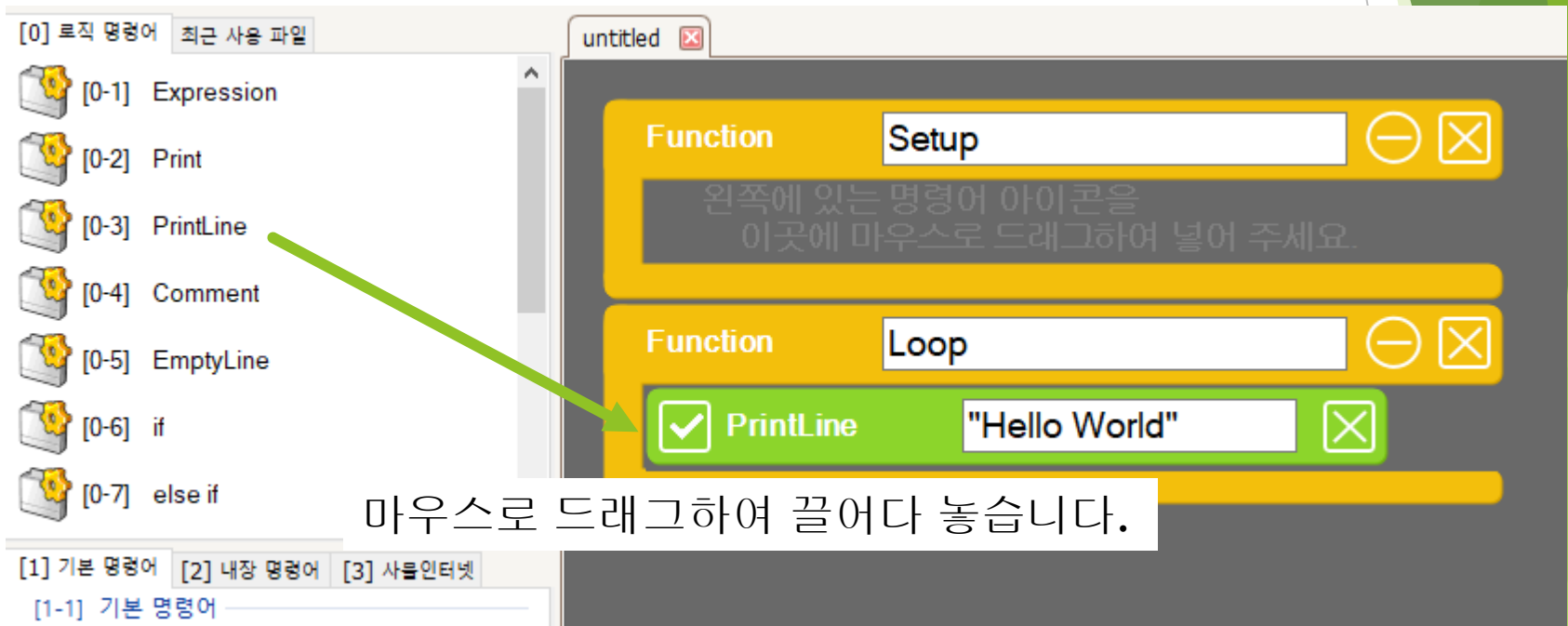
▶ 실습)

“Hello World” 단어를 1초 간격으로
줄을 바꾸어서 전송하기

아두이노 -> PC 값 전송하기

아두이노 통신

- ▶ [0-3] PrintLine 명령어를 추가합니다.
- ▶ PrintLine 입력창 안에 “Hello World”를 입력합니다.



The screenshot shows the Arduino IDE interface. On the left, the 'Blocks' palette is open, displaying various code blocks. A green arrow points from the 'PrintLine' block in the palette to the 'PrintLine' block in the 'Loop' function on the right. The 'PrintLine' block in the 'Loop' function has the text 'Hello World' entered in its input field. A text box at the bottom of the screenshot contains the instruction: '마우스로 드래그하여 끌어다 놓습니다.' (Drag with the mouse and drop it).

[0] 트릭 명령어 최근 사용 파일

[0-1] Expression

[0-2] Print

[0-3] PrintLine

[0-4] Comment

[0-5] EmptyLine

[0-6] if

[0-7] else if

[1] 기본 명령어 [2] 내장 명령어 [3] 사물인터넷

[1-1] 기본 명령어

untitled

Function Setup

원쪽에 있는 명령어 아이콘
이곳에 마우스로 드래그하여 넣어 주세요.

Function Loop

PrintLine "Hello World"

마우스로 드래그하여 끌어다 놓습니다.

아두이노 -> PC 값 전송하기

스크립트

- ▶ [0-3] PrintLine 명령어를 추가합니다.
- ▶ PrintLine 입력창 안에 “Hello World”를 입력합니다.

```
void setup()
{
}
void loop()
{
    PrintLine(“Hello World”)
}
```


아두이노 -> PC 값 전송하기

아두이노 통신

- ▶ [1-1-1] Delay 명령어를 PrintLine 명령어 아래에 추가합니다.

The screenshot shows the Arduino IDE interface. On the left, the 'Blocks' palette is open, showing a list of blocks under the 'Basic' category. The 'Delay' block is highlighted with a blue selection bar. A green arrow points from this 'Delay' block to the 'Delay' block in the 'Loop' function block in the main workspace. The 'Loop' function block contains a 'PrintLine' block with the text 'Hello World' and a 'Delay' block with the value '1000 (ms)'. The 'Setup' function block is empty. A text box in the workspace contains the instruction: '왼쪽에 있는 명령어 아이콘을 이곳에 마우스로 드래그하여 넣어 주세요.' (Drag the command icon from the left here with the mouse).

마우스로 드래그하여 끌어다 놓습니다.

아두이노 -> PC 값 전송하기

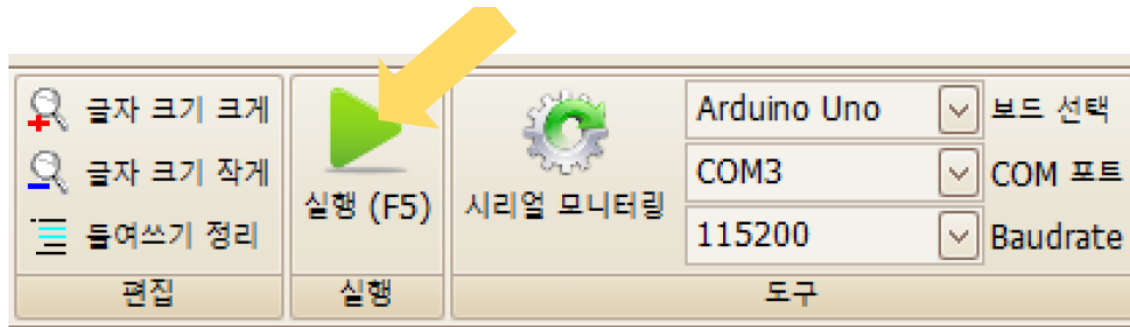
스크립트

- ▶ [1-1-1] Delay 명령어를 PrintLine 명령어 아래에 추가합니다.

```
void setup()
{
}
void loop()
{
    PrintLine("Hello World")
    Delay(1000)
}
```

아두이노 -> PC 값 전송하기

- ▶ 상단 메뉴의 가운데 있는 실행 버튼을 클릭하여 프로그램을 아두이노에 업로드 시킵니다.



실행 버튼을 클릭하여 프로그램을 아두이노 보드에 업로드 시킨다

프로그램을 실행한 후, 콘솔창에 표시되는 결과를 확인해 봅니다.

실습

▶ 실습)

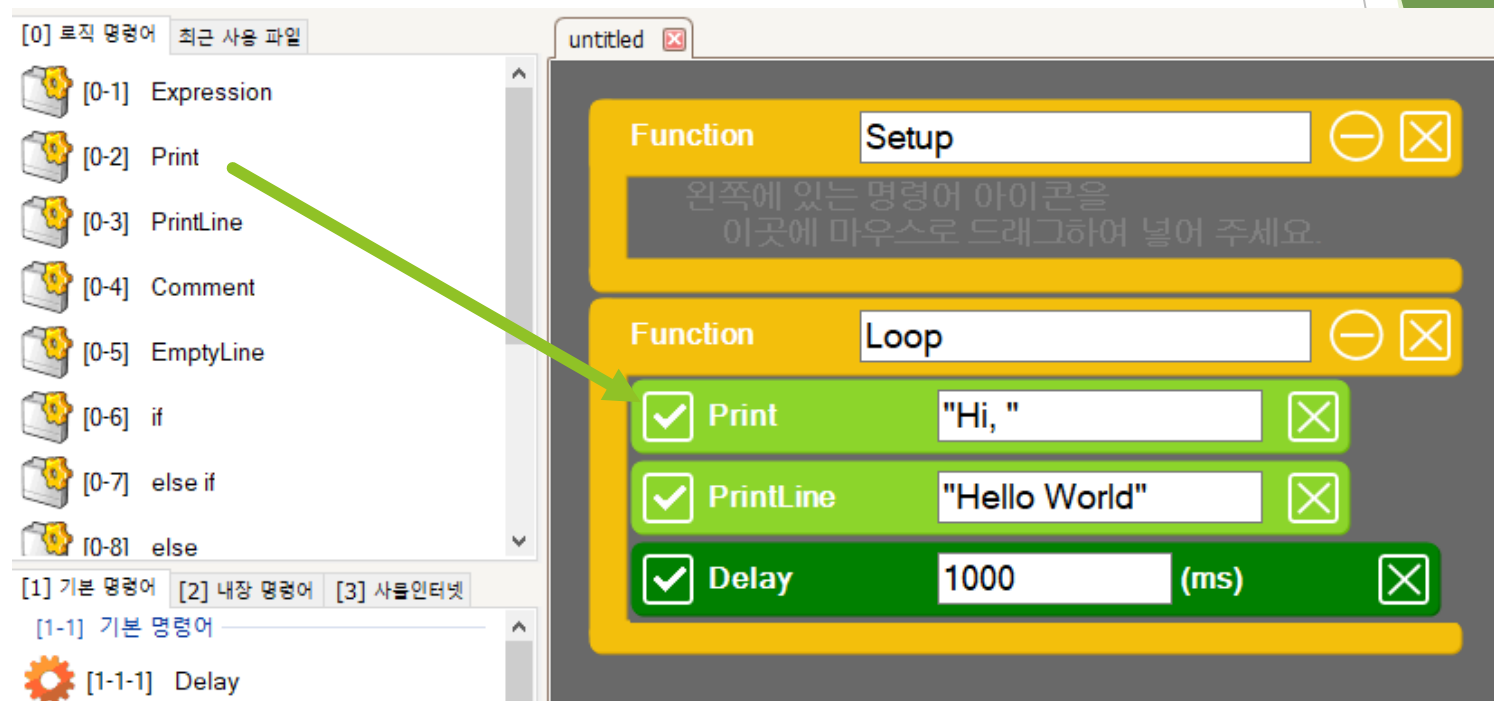
“Hi, Hello World” 단어를 1초 간격으로
줄을 바꾸어서 전송하기

단, “Hi, “를 먼저 출력한 후, 이어서 “Hello World”를
출력하여 2개의 문자열이 연결되도록 합니다.

아두이노 -> PC 값 전송하기

아두이노 통신

- ▶ [0-2] Print 명령어를 Loop 함수의 맨 위에 추가합니다.



마우스로 드래그하여 끌어다 놓습니다.

아두이노 -> PC 값 전송하기

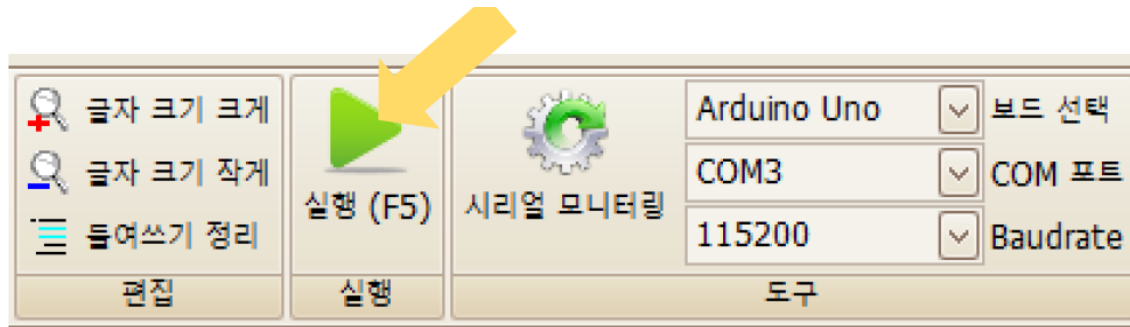
- ▶ [0-2] Print 명령어를 Loop 함수의 맨 위에 추가합니다.

```
void setup()
{

}
void loop()
{
    Print("Hi, ")
    PrintLine("Hello World")
    Delay(1000)
}
```

아두이노 -> PC 값 전송하기

- ▶ 상단 메뉴의 가운데 있는 실행 버튼을 클릭하여 프로그램을 아두이노에 업로드 시킵니다.



실행 버튼을 클릭하여 프로그램을 아두이노 보드에 업로드 시킨다

프로그램을 실행한 후, 콘솔창에 표시되는 결과를 확인해 봅니다.

디지털 버튼값 출력하기

디지털 버튼 값을 화면에 출력해 봅니다.

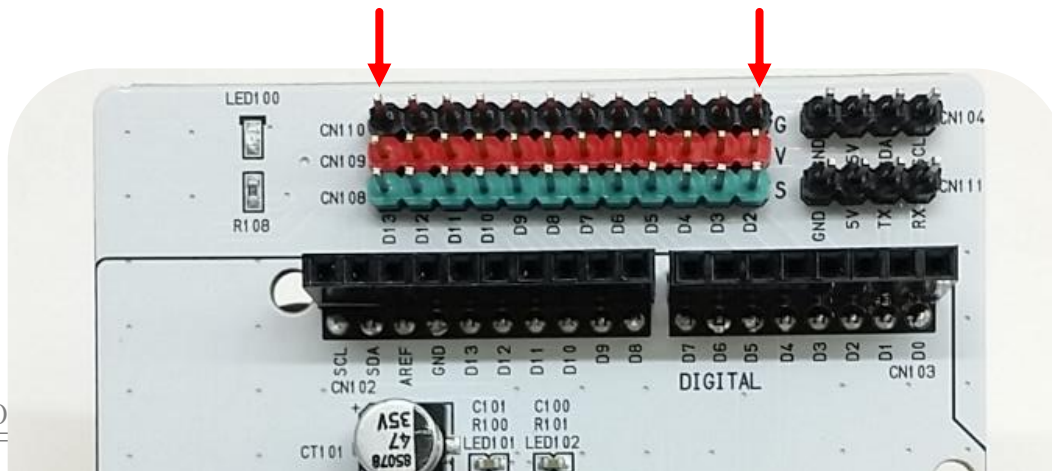
디지털 버튼 연결하기

- ▶ 다음과 같이 디지털 13번에 LED, 디지털 2번에 버튼을 연결해 봅니다.



13번

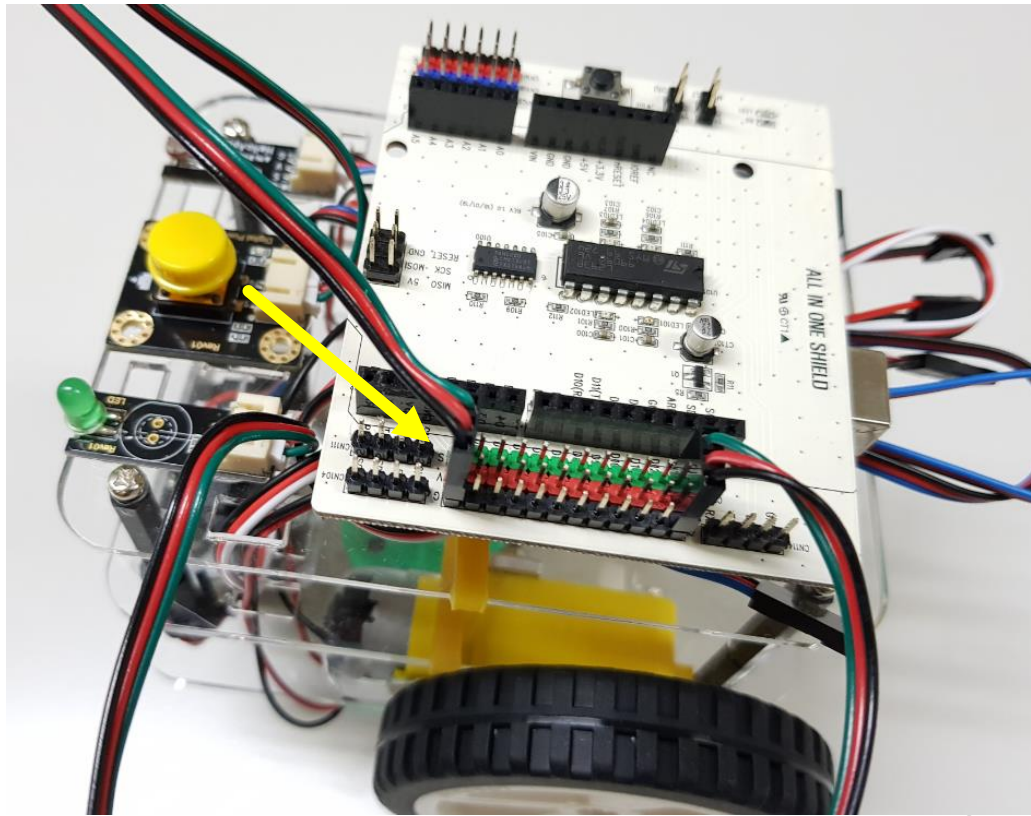
2번



디지털 버튼 연결하기

디지털 버튼

- ▶ 다음과 같이 디지털 13번에 LED, 디지털 2번에 버튼을 연결해 봅니다.



디지털 버튼 값 출력하기

▶ 디지털 버튼의 값 읽어오기

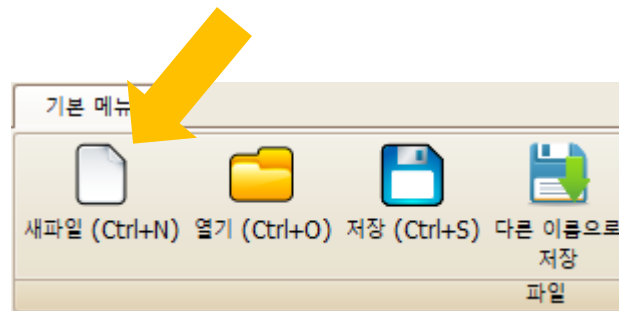
디지털 버튼 값을 읽어오기 위해서는 **DigitalRead** 명령어를 사용합니다.

`a = DigitalRead(핀번호)`



디지털 버튼 값 출력하기

- ▶ 왼쪽 맨 위에 있는 새파일 메뉴를 눌러 새로운 파일을 생성합니다.



디지털 버튼 값 출력하기

디지털 버튼

- ▶ [1-1-2] DigitalRead 명령어를 추가해 줍니다.

The screenshot shows a block-based programming environment with a left sidebar and a main workspace. The sidebar is organized into folders: [0] 트릭 명령어 (Trick commands), [0-1] Expression, and [1-1-2] Print. Under the [1-1-2] Print folder, there are sub-folders [1] 기본 명령어 (Basic commands), [2] 내장 명령어 (Built-in commands), and [3] 사물인터넷 (IoT). The [1-1] 기본 명령어 folder is expanded, showing blocks [1-1-1] Delay, [1-1-2] DigitalRead, [1-1-3] DigitalWrite, [1-1-4] DigitalWriteHigh, and [1-1-5] DigitalWriteLow. The [1-1-2] DigitalRead block is highlighted with a blue selection bar, and a green arrow points from it to the workspace. The workspace contains two function blocks: 'Function Setup' and 'Function Loop'. The 'Function Loop' block has a sub-block with a checked checkbox, a variable 'd', an equals sign, the text 'DigitalRead', a value '2', and '(Pin)'. A grey text box above the 'Function Loop' block contains the Korean text: '왼쪽에 있는 명령어 아이콘을 이곳에 마우스로 드래그하여 넣어 주세요.' (Please drag the command icon on the left here with the mouse).

디지털 버튼 값 출력하기

스크립트

- ▶ [1-1-2] DigitalRead 명령어를 추가해 줍니다.

```
void setup()
{

}
void loop()
{
    d = DigitalRead(2)
}
```

디지털 버튼 값 출력하기

디지털 버튼

- ▶ [0-3] PrintLine 명령어를 추가해 줍니다.

The screenshot shows a block-based programming interface. On the left, a '최근 사용 파일' (Recent Files) panel lists several blocks: [0-1] Expression, [0-2] Print, [0-3] PrintLine (highlighted in blue), [0-4] Comment, and [0-51] EmotvLine. Below this, a '기본 명령어' (Basic Blocks) panel shows [1-1] 기본 명령어 (Basic Blocks) with sub-blocks [1-1-1] Delay, [1-1-2] DigitalRead, and [1-1-3] DigitalWrite. The main workspace, titled 'untitled', contains two function blocks: 'Function Setup' and 'Function Loop'. The 'Function Loop' block contains two sub-blocks: an orange 'DigitalRead' block with 'd' in a variable field and '2 (Pin)', and a green 'PrintLine' block with 'd' in a variable field. A green arrow points from the 'PrintLine' block in the left panel to the 'PrintLine' block in the 'Function Loop' block. A grey text box in the 'Function Setup' block contains the instruction: '왼쪽에 있는 명령어 아이콘을 이곳에 마우스로 드래그하여 넣어 주세요.' (Drag the command icon from the left here with the mouse).

디지털 버튼 값 출력하기

스크립트

- ▶ [0-3] PrintLine 명령어를 추가해 줍니다.

```
void setup()
{

}
void loop()
{
    d = DigitalRead(2)
    PrintLine(d)
}
```

디지털 버튼 값 출력하기

디지털 버튼

- ▶ [1-1-1] Delay 명령어를 추가한 후, 값을 100으로 수정해 줍니다.

The screenshot shows a programming environment with a left sidebar and a main workspace. The sidebar is divided into three sections: [0] 트릭 명령어 (Trick commands), [1] 기본 명령어 (Basic commands), and [2] 내장 명령어 (Built-in commands). Under [1-1] 기본 명령어, the 'Delay' block is highlighted with a blue selection bar. A green arrow points from this block to the workspace. The workspace shows a 'Loop' function block containing three sub-blocks: 'DigitalRead' (pin 2), 'PrintLine' (variable 'd'), and 'Delay' (100 ms). A yellow arrow points to the '100' value in the 'Delay' block. The 'Setup' function block is also visible above the 'Loop' block.

디지털 버튼 값 출력하기

스크립트

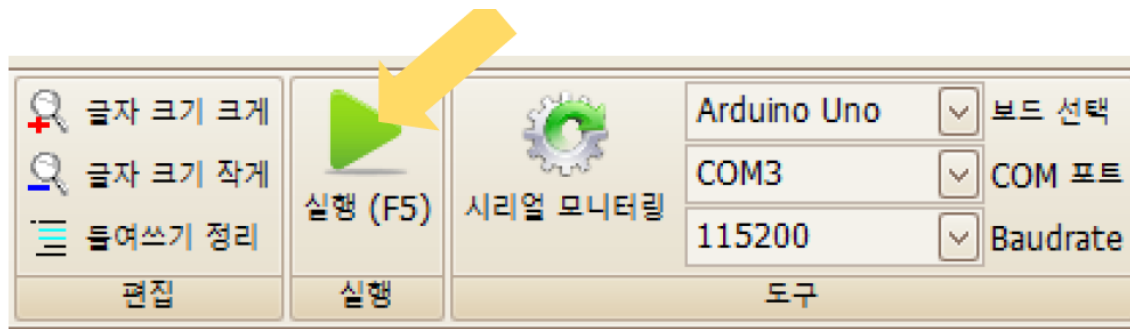
- ▶ [1-1-1] Delay 명령어를 추가한 후, 값을 100으로 수정해 줍니다.

```
void setup()
{

}
void loop()
{
    d = DigitalRead(2)
    PrintLine(d)
    Delay(100)
}
```

디지털 버튼 값 출력하기

- ▶ 상단 메뉴의 가운데 있는 실행 버튼을 클릭하여 프로그램을 아두이노에 업로드 시킵니다.



실행 버튼을 클릭하여 프로그램을 아두이노 보드에 업로드 시킨다



버튼을 눌러 가면서 콘솔창에 표시되는 결과를 확인해 봅니다.

버튼이 눌러지면 LED 켜기

버튼이 눌러지면 LED 켜기

디지털 버튼

▶ 실습 목표

- ▶ 버튼을 누르고 있는 동안 LED가 켜지게 합니다.

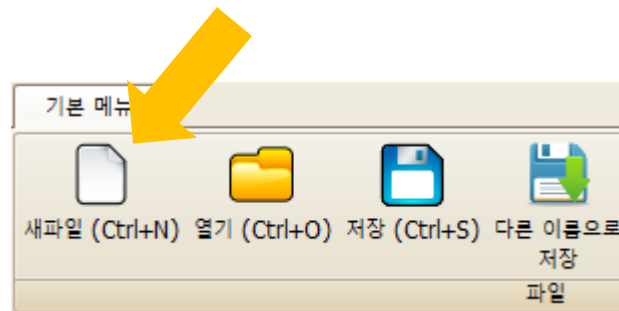
▶ 실습 내용

- ▶ 디지털 버튼의 눌림 상태를 읽어 옵니다.
- ▶ 디지털 버튼의 상태에 따라 다음과 같이 LED를 제어합니다.
 - ▶ 버튼이 눌러져 있으면 LED를 켭니다.
 - ▶ 버튼이 눌러져 있지 않으면 LED를 끕니다.

버튼이 눌러지면 LED 켜기

디지털 버튼

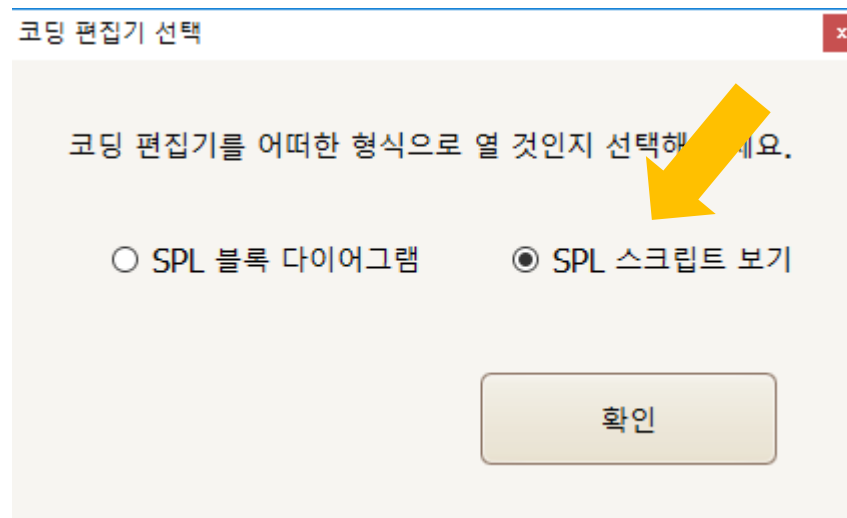
- ▶ 왼쪽 맨 위에 있는 새파일 메뉴를 눌러 새로운 파일을 생성합니다.



버튼이 눌러지면 LED 켜기

디지털 버튼

- ▶ 왼쪽 맨 위에 있는 새파일 메뉴를 눌러 새로운 파일을 생성합니다.

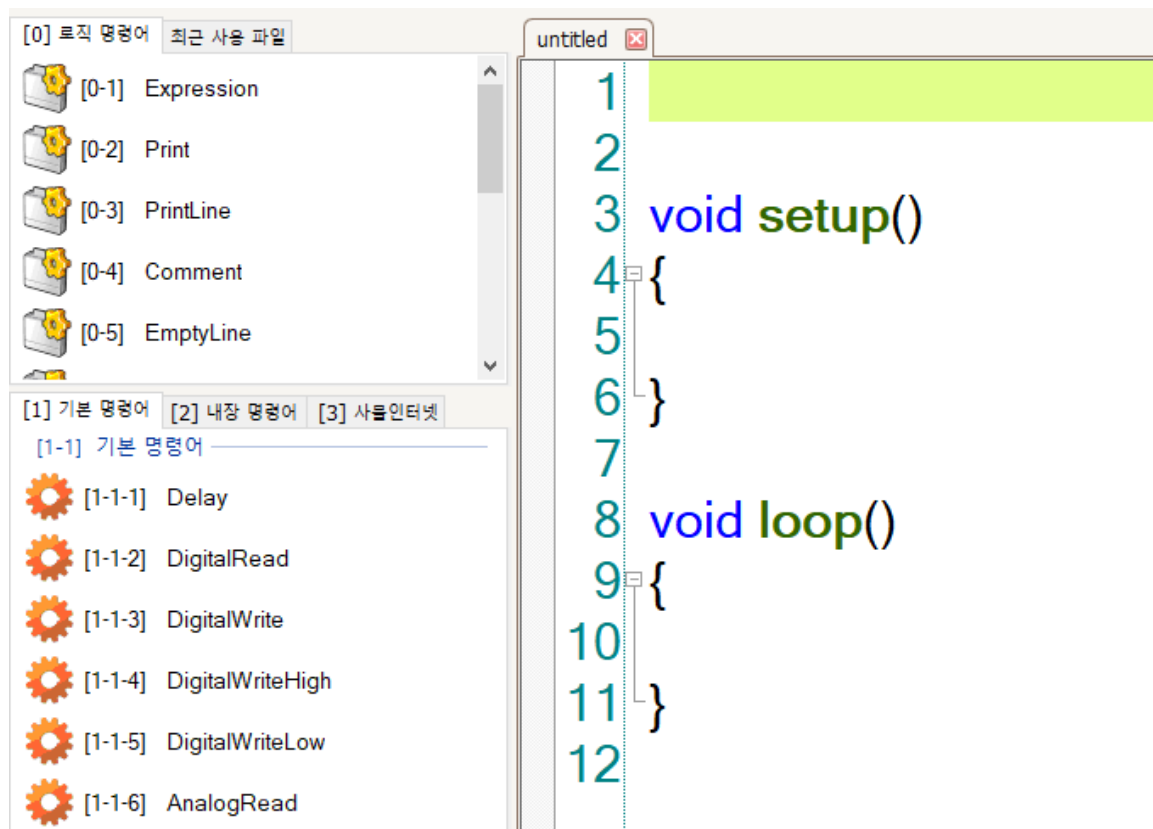


스크립트 형식의 편집기를 선택합니다.

버튼이 눌려지면 LED 켜기

디지털 버튼

- ▶ 스크립트 방식의 기본 편집기 창



버튼이 눌러지면 LED 켜기

- ▶ 버튼 센서값을 읽어 온 후 출력하는 예제를 스크립트로 작성해 봅니다.
- ▶ 기존에 저장된 예제를 스크립트 형식으로 열어도 됩니다.

```
void setup()
{

}

void loop()
{
    d = DigitalRead(2)
    PrintLine(d)
    Delay(100)
}
```

버튼이 눌러지면 LED 켜기

- ▶ if 조건 비교 및 else 조건 구문을 다음과 같이 추가합니다.

```
void loop()
{
    d = DigitalRead(2)
    PrintLine(d)

    if (d == HIGH)
    {

    }
    else
    {

    }
    Delay(100)
}
```

버튼이 눌러지면 LED 켜기

- ▶ 프로그램 기능을 완성해 봅니다.

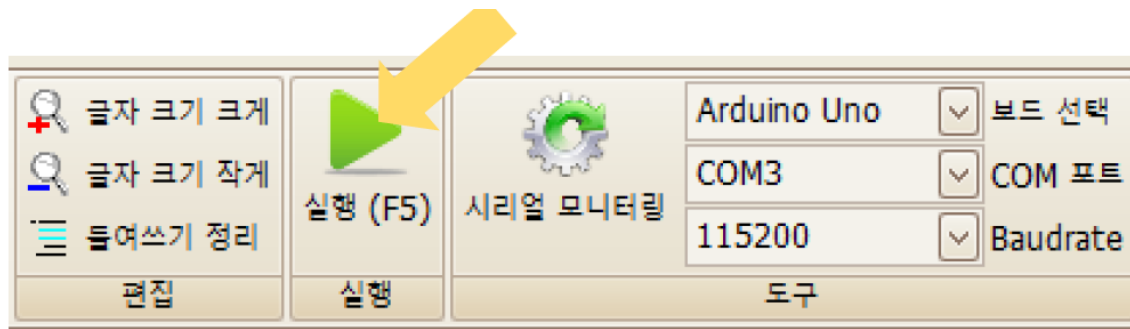
```
void loop()
{
    d = DigitalRead(2)
    PrintLine(d)

    if (d == HIGH)
    {
        DigitalWrite(13, HIGH)
    }
    else
    {
        DigitalWrite(13, LOW)
    }
    Delay(100)
}
```

버튼이 눌러지면 LED 켜기

디지털 버튼

- ▶ 상단 메뉴의 가운데 있는 실행 버튼을 클릭하여 프로그램을 아두이노에 업로드 시킵니다.



실행 버튼을 클릭하여 프로그램을 아두이노 보드에 업로드 시킨다



버튼을 눌러서 LED가 켜지는 지 관찰합니다.

버튼과 LED 활용 실습

▶ 실습 목표

- ▶ 주제1) 토글 버튼으로 **LED** 제어하기
- ▶ 주제2) 반복 놀림을 방지하기 위한 기능 구현하기
- ▶ 주제3) 버튼으로 숫자 세는 장치 만들기
- ▶ 주제4) 눌러진 숫자를 **LED**로 횡수 표시하기
- ▶ 주제5) 버튼을 **2초간** 누르고 있으면 기능 초기화 하기

주제1) 토글 버튼으로 LED 켜기

실습

- ▶ 실습 목표
 - ▶ 디지털 버튼으로 토글 기능을 구현해 봅니다.
- ▶ 실습 내용
 - ▶ 전역변수로 토글 상태를 저장합니다.
 - ▶ 버튼이 눌러지면 토글 상태를 변경합니다.
 - ▶ 토글 상태에 따라 **LED**를 켜거나 끕니다.

주제1) 토글 버튼으로 LED 켜기

실습

```
togle = false
```

```
void loop()  
{
```

```
    d4 = DigitalRead(4)
```

```
    if (d4 == HIGH)
```

```
    {
```

```
        if (togle)
```

```
            togle = false
```

```
        else
```

```
            togle = true
```

```
    }
```

```
    PrintLine(togle)
```

```
    if (togle)
```

```
        DigitalWrite(13, HIGH)
```

```
    else
```

```
        DigitalWrite(13, LOW)
```

```
    Delay(100)
```

```
}
```

The screenshot shows a block-based programming environment with the following structure:

- Expression:** togle = false
- Function:** Loop
- Block:** d4 = DigitalRead 4 (핀번호)
- if:** d4 == HIGH
 - if:** togle
 - Expression:** togle = false
 - else:**
 - Expression:** togle = true
- PrintLine:** togle
- Delay:** 100 (밀리초)

주제2) 키 중복눌림 방지하기

실습

▶ 실습 목표

- ▶ 전역 변수로 키 중복 눌림을 방지시켜 봅니다.

▶ 실습 내용

- ▶ 전역변수로 토글 상태를 저장합니다.
- ▶ 전역변수로 버튼이 눌러져 있는지 체크합니다.
- ▶ 버튼이 최초 눌러질 때에만 토글 상태를 변경합니다.
- ▶ 토글 상태에 따라 **LED**를 켜거나 끕니다.

주제2) 키 중복눌림 방지하기

실습

```
togle = false  
pressed = false
```

전역변수

```
void loop()  
{  
    d4 = DigitalRead(4)  
    if (d4 == HIGH)  
    {  
        if (pressed == false)  
        {  
            if (togle)  
                togle = false  
            else  
                togle = true  
        }  
        pressed = true  
    }  
    else  
        pressed = false  
  
    PrintLine(togle)  
  
    if (togle)  
        DigitalWrite(13, HIGH)  
    else  
        DigitalWrite(13, LOW)  
  
    Delay(100)  
}
```

주제3) 버튼으로 숫자 세기

실습

▶ 실습 목표

- ▶ 키 중복 눌림을 방지 기능을 활용하여 버튼의 눌러진 횟수를 화면에 표시합니다.

▶ 실습 내용

- ▶ 전역변수로 버튼이 눌러져 있는지 체크합니다.
- ▶ 전역변수로 눌러진 횟수를 저장하는 변수를 정의합니다.
- ▶ 버튼이 최초 눌러질 때에만 횟수 변수의 값을 증가시키고 화면에 값을 출력시킵니다.

주제3) 버튼으로 숫자 세기

실습

```
count = 0
pressed = false

void loop()
{
    d4 = DigitalRead(4)
    if (d4 == HIGH)
    {
        if (pressed == false)
        {
            count = count + 1
            PrintLine(count)
        }
        pressed = true
    }
    else
        pressed = false

    Delay(100)
}
```

주제4) 눌러진 숫자를 LED 점멸로 표시하기

실습

▶ 실습 목표

- ▶ 눌러진 횟수를 LED 점멸로 표시해 봅니다.

▶ 실습 내용

- ▶ 전역변수로 버튼이 눌러져 있는지 체크합니다.
- ▶ 전역변수로 눌러진 횟수를 저장하는 변수를 정의합니다.
- ▶ 버튼이 최초 눌러질 때에만 횟수 변수의 값을 증가시키고 화면에 값을 출력시킵니다.
- ▶ 눌러진 횟수 만큼 LED를 0.1초 간격으로 점멸 시켜 봅니다.

주제4) 눌러진 숫자를 LED 점멸로 표시하기

실습

```
count = 0
pressed = false

void loop()
{
    d4 = DigitalRead(4)
    if (d4 == HIGH)
    {
        if (pressed == false)
        {
            count = count + 1
            for (i = 0; i < count; i++)
            {
                DigitalWrite(2, HIGH)
                delay(100)
                DigitalWrite(2, LOW)
                delay(100)
            }
        }
        pressed = true
    }
    else
        pressed = false

    Delay(100)
}
```

주제5) 2초간 누르면 기능 초기화 하기

실습

▶ 실습 목표

- ▶ 기존에 만들어진 기능에 초기화 기능을 추가해 봅니다.

▶ 실습 내용

- ▶ 전역변수로 버튼이 눌러져 있는지 체크합니다.
- ▶ 전역변수로 눌러진 횟수를 저장하는 변수를 정의합니다.
- ▶ 버튼이 눌러져 있는 시간을 저장하는 전역변수를 정의합니다.
- ▶ 버튼이 최초 눌러질 때에만 횟수 변수의 값을 증가시키고 화면에 값을 출력시킵니다.
- ▶ 버튼이 눌러져 있는 동안에는 시간 변수의 값을 증가시킵니다.
- ▶ 시간 변수 값이 **10**을 넘으면 횟수 변수의 값을 **0**으로 초기화 합니다.

주제5) 2초간 누르면 기능 초기화 하기

실습

```
count = 0
pressed = false
t = 0

void loop()
{
    d4 = DigitalRead(4)
    if (d4 == HIGH)
    {
        if (pressed == false)
        {
            count = count + 1

            for (i = 0; i < count; i++)
            {
                DigitalWrite(2, HIGH)
                delay(100)
                DigitalWrite(2, LOW)
                delay(100)
            }

            pressed = true
            t = t + 1
        }
    }
}
```

```
else
{
    pressed = false
    if (t > 10)
        count = 0
    t = 0
}
Delay(100)
}
```