GPIO

이 장에서는 마이크로 컨트롤러의 GPIO를 다루어서 제어하는 방법에 대해 이야기 해본다.

GPIO를 통한 제어는 일반적인 기기에서 스위치를 통해 제어하는것과 같은 가장 기초적인 제어방식이다. 따라서 익숙해져야 될 필요가 있다. 마이크로 컨트롤러 에서의 제어는 일반 컴퓨터와 비슷하게 동작하는데 GPIO의 몃번핀에 연결된 메모리에 HI값과 LOW값을 넣어서 제어하는 식이다. 따라서 모든 레지스터의 메모리의 물리주소를 다 외울 수 있다면 메모리 주소만 가지고도 프로그래밍을 할수 있겟지만 이러면 대부분의 사람들은 코드를 봐도 이해를 할 수가 없기 때문에 일반적으로 쓰이는 방식에 대해 이야기 해보겟다.

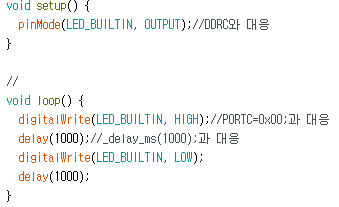
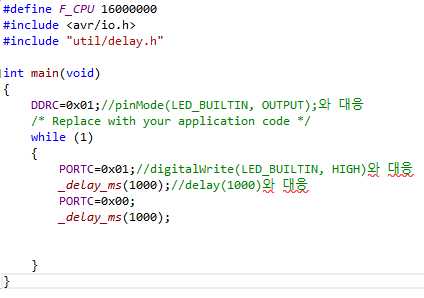
기본적으로 같은 마이크로 컨트롤러를 사용하는 아두이노와 AVR의 코드는 일종의 유사성을 지니고 있다.

그림1 AVR과 아두이노의 GPIO제어코드 예제.

기본적으로 AVR의 GPIO를 제어하는 과정은 보통 이렇다.[[1]](#footnote-1)

1. DDR# 레지스터에 자신이 원하는 출력 방향을 지정해준다 입력값이 1일경우 출력 방향이며 0일경우 입력방향이다.
2. PORT# 레지스터에 자신이 원하는 출력값을 넣어준다.

예시로 올린 AVR코드에서는 기독성을 위해 16진수를 사용했으나 사용하기에 따라서 10진수나 식별의 편의성을 위해 2진수를 사용해도 별 문제가 없다.

AVR코드를 처음 본 사람은 저게 어떤 핀을 제어하는 것 인지 처음에는 이해하지 못할수도 있다. 이제부터 AVR코드가 어떤 의미를 지니고 있는지 확인해 보자. 기본적으로 8비트 AVR은 GPIO레지스터 구분이 8비트로 구성되어 8핀으로 되어있다. 즉 PORT#는 8비트로 구성된 제어 레지스터로 내부에 8비트 데이터를 기록하는 것으로 제어가 가능하다. 이경우 위의 PORTC=0x01을 2진수로 표현할 경우 0b00000001[[2]](#footnote-2)식으로 표현되며 이는

PORTC 레지스터

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| PC7핀 | PC6핀 | PC5핀 | PC4핀 | PC3핀 | PC2핀 | PC1핀 | PC1핀 |

이런 식으로 표현 할수있다. 실제로도 0b00000001로 사용해도 되지만 보통은 짧고 사용성도 나쁘지 않은 16진수로 기술하는 방식이 가장 많이 쓰인다.

입력도 이와 비슷하게 쓰인다. 출력을 담당하는 레지스터가 있듯이 입력값을 받는 레지스터 또한 존재한다. 입력을 받는 레지스터는 PIN#로 쓰이며 읽어온 핀 데이터를 8비트 데이터로 반환한다. 따라서 일반적으로 데이터를 받아오면 8비트 자료형 숫자값으로 표현되어 나오는데 이경우 입력을 알아보면.

PINC 레지스터

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| PC7핀 | PC6핀 | PC5핀 | PC4핀 | PC3핀 | PC2핀 | PC1핀 | PC1핀 |

만약 PC7입력이 이런 상태일 경우 PINC레지스터에는 0b00110110이라는 데이터가 들어가고 이값을 그대로 10진수 정수로 읽어올 경우 54란 값으로 나온다. 그러면 우리는 이 값을 통해 1,2,4,5번핀이 HIGH입력이 들어왔다고 판단할 수 있는 것이다.

코드는 PORTC의 0번 핀을 1로 만든뒤 다시 0으로 만드는걸 무한 반복하는 코드이다.

1. #은 사용할 GPIO에 따라 달라질수 있는 자리다. 위의 예시로 올라온 코드에서는 C가 쓰였기 떄문에 DDRC,PORTC라고 서술되어있다. [↑](#footnote-ref-1)
2. 프로그래밍언어에서의 타 진법 표기방식이다 0x에서 x는 16진법라는 뜻의 hexadecimal 의 x고 0b 의 b 이진법을 뜻하는 binary의 b를 뜻한다 [↑](#footnote-ref-2)