|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **예 비 보 고 서** | | | | |
| 학 과 | 학 년 | 학 번 | 조 | 성 명 |
| 전자공학과 | 3 | 12191505 | - | 윤수연 |
| 실험 제목 | GPIO | | | |
| 1. **자료조사**    1. **ATmega128이란**   ATmega128은 128bit로 동작하는 마이크로 프로세서 중 한 종류다. 아래 사진이 바로 ATmega128의 실제 사진이다. 가운데에 큰 칩이 하나 있고 칩의 각 핀은 여러 부품과 연결되어 있다. 모든 부품은 큰 보드 위에 있는데 사실 마이크로 컨트롤러 그 자체는 가운데의 칩을 말하고 보드는 단지 편의성을 위한 확장 부품이라고 보면 된다. 핀에 직접 접근하여 모든 동작을 수행하기에 어렵기 때문에 보드 위에 올린 구조다. 이렇게 여러 기능을 보드 하나로 수행할 수 있다.  Atmega 128 atmega128 avr 최소 코어 시스템 개발 보드 모듈 isp for arduino|Instrument  Parts &amp; Accessories| - AliExpress   * 1. **ATmega128의 마이크로 컨트롤러 구조**   위에서 언급했듯이 마이크로 컨트롤러는 작게는 칩 하나만을 말한다. 이때 칩을 위에서 자세히 살펴보면 정사각형의 모양으로 각 모서리에 여러 개의 핀이 달려있다. 각 칩에 대한 핀은 모두 다르지만 ATmega128의 경우 각 모서리에 모두 동일하게 16개의 핀이 달려있는데 4면이므로 이 마이크로 컨트롤러에는 총 64개의 핀이 달려있는 셈이다. 아래 사진은 각 핀의 이름을 표시한 사진이다. P로 시작하는 이름의 경우 P가 Port의 약자로 port역할을 하는 것이다. 이 중에 입출력을 담당하는 I/O 포트의 경우 64개 중 53개를 차지하는데 A포트부터 G포트까지 총 7개의 포트종류로 나뉜다. 이에 따라 포트에 부여되는 비트 크기가 다르다. A부터 F까지는 8비트로 이루어져 있고 G포트의 경우 5비ㅡㅌ로 이루어져 있다.  AVR(ATmega128)의 핀 구조 및 핀 설명   * 1. **GPIO란?**   비트가 다른 것처럼 각각 다른 역할로 쓰일 수 있게 다중기능을 가지고 있는데 I/O포트인 만큼 기본적으로 입출력 기능은 동일하다. 이 때의 입출력 기능을 GPIO라고 하며, ATmega128의 I/O포트는 모두 GPIO기능을 가지고 있다. GPIO는 General Purpose Input/Output의 약자다. 즉 이름 그대로 Input기능과 output기능을 말한다.  마이크로 컨트롤러에서 input/output신호는 0또는 1로 표현되는데 이를 아날로그에서는 전압의 크기로 이해할 수 있다. 전압이 0V면 off, 즉 0으로, 전압이 5V로 주어지면 on, 즉 1로이해하는 것이다. 이러한 방식으로 GPIO는 데이터를 읽고 쓸 수 있으며 전원을 on/off할 수 있다.  그리고 각 I/O포트는 아래 그림과 같은 회로구성을 갖는데, 여기서 구조를 크게 세 가지로 볼 수 있다. 첫번째로 다이오드다. 다이오드는 간단히 말하면 특정 전압이상의 값만 통과시키는 소자로, 포트의 입력으로 너무 크거나 작은 값이 들어왔을 때 이를 저지하는 역할을 한다. 즉 극단적인 값들로부터 회로를 보호한다.  다음으로 풀-업 저항이다. 입력이 0이나 1로 주어지는 것이 아니라 아무것도 주어지지 않는 경우 값을 인식하지 못하고 입력이 파르르 떨리는 현상을 floating현상이라고 하는데 이러한 현상을 저항을 달아주어 억제한다. 즉 디폴트 값을 정해주는 역할이라고도 생각할 수 있다. 입력 포트에서 필수다.  마지막으로 FET이 달려있다. 이것은 내부적으로 스위치 역할을 하는데, on/off로 풀-업 저항을 사용할 것인지, 사용하지 않을 것인지 정한다. 이를 풀-업저항을 인에이블한다고 말한다. 이 외에도 capacitor 등 여러 소자가 있다.  AVR(Atmega, Mega board) 배우기: I/O Port의 이해와 프로그래밍: Switch, LED   * 1. **GPIO의 사용**   ATmega128의 경우 모든 제어가 프로그램 상 레지스터 설정을 통해 가능하다. 이 때 레지스터란 임시저장소라고 생각하면 된다. 즉 프로그램코드로 사전에 레지스터를 정의하고 그 값을 설정함으로써 동작을 제어하는 것이다. 우리가 GPIO를 위해 사용하는 레지스터는 총 세가지가 있다.   * + 1. **DDR레지스터**   DDR레지스터는 해당 포트에서 데이터를 입력으로 받아들일 것인지 출력으로 내보낼 것인지 결정한다. 해당 포트 위치의 값이 0일 경우 입력, 1일 경우 출력을 담당한다.   * + 1. **PORT레지스터**   PORT 레지스터의 경우 해당 포트가 입력을 담당하는지 출력을 담당하는지에 따라 역할이 달라진다. 만약 입력의 경우 해당 포트 위치에 값이 1이면 풀업저항을 on, 즉 FET을 on 시키고, 0이면 그 반대로 FET을 off시킨다. 반면에 출력의 경우 진짜 그 출력 값을 내보낸다고 생각하면 된다. 해당 포트 위치의 값이 1이라면 1, 즉 5V를 내보내고 0이라면 0, 즉 0V를 내보낸다.   * + 1. **PIN레지스터**   PIN레지스터는 특이하게 DDR레지스터 단계에서 해당 포트를 입력으로 설정했을 때만 사용한다. 해당 포트의 입력 값을 읽어서 저장하는 임시저장소다. input변수에 저장한다.   1. **실험**    1. **연산 활용**   ATmega128의 GPIO는 2진수로 이루어진다. 따라서 비트단위로 논리연산을 수행하는 비트연산이 가능하다. 이를 이용하여 대표적인 비트 연산자 OR와 AND를 이용하여 원하는 핀에만 접근하는 것이 가능하다. 이러한 연산의 활용은 마이크로 컨트롤러를 보다 자유롭게 이용할 수 있게 해준다.   * + 1. **OR**   표준 논리게이트 NAND 게이트, NOR 게이트   * + 1. **AND**   표준 논리게이트 NAND 게이트, NOR 게이트   * 1. **실험 예상**      1. **실습 1단계**   1개의 버튼 입력이 들어올 때마다 1개의 LED on/off의 반복구현: led on -> button누름 ->led off, led off -> button누름 -> led on  입력과 출력 port가 각각 하나씩 기능하여 각 기능에 port 하나씩을 부여하면 될 것 같다. 코드 작성은 버튼에 따라 led의 on off가 결정되는 내용의 코드를 작성하면 될 것 같고 button소자의 사용이 처음이라 이 부분이 어려울 것 같다.   * + 1. **실습 2단계**   3개의 스위치(1: on/off, 2: speed up, 3: speed down)와 8개의 led가 존재하는 상황에서 스위치 입력에 따른 led조작: led는 순서대로 불이 들어왔다 꺼지는 과정이 반복, led가 off->on이 되면 마지막에 동작했던 speed로 led동작  3개의 입력과 8개의 출력으로 1단계보다 port사용 수가 많아졌고 디폴트 동작이led 순서대로 불이 들어왔다 꺼지는 동작이라 이 부분이 어려울 것 같다. 그리고 led가 off->on이 되면 마지막에 동작했던 speed로 led동작하려면 speed가 기억되어야 할 것이다.   * + 1. **실습 3단계**   세븐 세그먼트를 연결하여 4개 스위치 입력에 따른 숫자 변화: 4개의 스위치 입력이 2진수로 표현이 되어 0~F까지 표현  세븐 세그먼트는 디지털 논리회로에서도 종종 본 익숙한 알고리즘이지만 우리가 인식하는 숫자와 그저 스위치로 조종하는 2진수 숫자와는 차이가 있어 복잡할 것 같다. 필요한 입력 수는 스위치 4개, 출력 수는 세븐 세그먼트이니 7개로 역시 복잡한 구조일 것이다. | | | | |
|  | | | | |