|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **결 과 보 고 서** | | | | |
| 학 과 | 학 년 | 학 번 | 조 | 성 명 |
| 전자공학과 | 3 | 12191505 | - | 윤수연 |
| 실험 제목 | Stepping Motor | | | |
| 1. **실험 과정**    1. 실습 1단계  |  | | --- | | #include <avr/io.h>  #include <util/delay.h>    int main(){  DDRE = 0xff ;  PORTE = 0x33;  //00110011 init  while(1){  PORTE = 0x66;  //01100110  \_delay\_ms(200);  PORTE = 0xcc;  //11001100  \_delay\_ms(200);  PORTE = 0x99;  //10011001  \_delay\_ms(200);  PORTE = 0x33;  //00110011  \_delay\_ms(200);  }    return 0 ;  } |       그림 . 실습 1단계 회로 구성도  실습 1단계는 2상 여자 방식으로 stepping motor를 360도 회전하는 프로그램을 작성하는 실험이다. 방향이나 속도 제어 없이 그저 회전하는 동작을 구현하는 것이라 코드도 간결하다. 실제 실험에서는 12V 전압으로 ATmega128 MCU와 Motor사이에 모터 컨트롤러라는 소자가 들어가서 모터 제어를 더 쉽게 해주지만, 하드웨어 가상 프로그램 Proteus에서는 ATmega128로 제어하는 것이나 컨트롤러로 제어하는 것이나 큰 차이가 없기 때문에 바로 직접 연결해 주었다.  본격적으로 코드를 보다 쉽게 동작을 이해하기 위해 주석처리한 부분을 참고하여 살펴 보면 2상 여자 방식을 구현했음을 알 수 있다. 한 바퀴를 구성하는 주기가 while문 안에서 반복되는 것을 확인할 수 있다. 이 때 0과 1의 비율이 50:50으로 2상 여자 방식이라는 것을 알 수 있다. Delay가 200ms로 우리 눈에도 감지되는 뚝뚝 끊기는 동작을 볼 수 있다.   * 1. 실습 2단계  |  | | --- | | #define F\_CPU 16000000  #define \_\_DELAY\_BACKWARD\_COMPATIBLE\_\_  #include <avr/io.h>  #include <util/delay.h>  int delay\_speed = 30; // init speed  unsigned char direction ; //direction = 1 -> CCW  void rotation(){ // function of motor moving by direction  if(direction == 1){ //CCW  PORTE = 0x66; //01100110  \_delay\_ms(delay\_speed);  PORTE = 0xcc; //11001100  \_delay\_ms(delay\_speed);  PORTE = 0x99; //10011001  \_delay\_ms(delay\_speed);  PORTE = 0x33; //00110011  \_delay\_ms(delay\_speed);  }  else{ // CW  PORTE = 0x66;  \_delay\_ms(delay\_speed);  PORTE = 0x33;  \_delay\_ms(delay\_speed);  PORTE = 0x99;  \_delay\_ms(delay\_speed);  PORTE = 0xcc;  \_delay\_ms(delay\_speed);  }  }  void direction\_ctrl(){ // change direction  if(direction==1)  direction = 0;  else  direction = 1;  }  int main(){  unsigned char key;    DDRD = 0x00 ; // input  PORTD = 0b00000111; // pull up resistance    DDRE = 0xff; // output  PORTE = 0x33;    direction = 1; // init direction = 1    while(1){  rotation();  key = (PIND & 0b00000111);  //\_delay\_ms(100);    switch(key) {  case 0b00000110: // button no.1  if(delay\_speed >50){  delay\_speed -= 10; // faster speed  }  break;    case 0b00000101: // button no. 2  delay\_speed +=10; // slower speedd  break;    case 0b00000011: //button no.3  direction\_ctrl(); // direction control(change)  break;    default: // exception  break;  }  }  } |   실습 2단계는 스위치 입력에 따라 스태핑 모터를 다양하게 제어할 수 있도록 구현하는 단계다. 세 개의 스위치를 통해 각각 회전 속도 증가, 감소, 그리고 방향 변환의 기능을 구현하면 된다.  그런데 지난주에서도 언급했지만 switch소자의 경우 on off상태가 유지된다는 점에서 button과 차이점이 있다. 예를 들어 on에서 off로 스위치를 조작했다면 그 스위치는 조작 이후에도 계속해서 off상태를 유지하는 것이 된다. 그에 비해 button의 경우 trigger를 무엇에 두느냐에 따라 다르지만 스위치를 누르는 순간 어떤 동작이 인식되고, 이후에는 다시 원래로 돌아온다. 따라서 속도 조절 등의 연속적인 동작에는 button소자를 사용하는 것이 올바른 동작일 것이다.  순서대로 회전 속도 증가를 control하는 button을 1번 button, 회전 속도 감소를 control하는 button을 2번 button, 마지막으로 방향을 control하는 button을 3번 button이라고 하고 코드를 작성했다. 모두 2상 여자 기반으로 동작한다. 따라서 역시 제어부분을 제외하면 실험 1단계와 유사하다.  다만 제어 부분이 추가된 형태다. 제어부는 앞서 말한 이유로 button으로 동작하도록 구현했다. Button의 입력을 감지하는 방법에는 크게 두 가지가 있는데 먼저 첫번째는 while반복문을 이용하여 매번 입력을 감지하고 이 값을 각 case에 맞게 동작하는 것이다. 실험 초기에 많이 사용한 방법이다.  두번째는 interrupt를 이용하는 것이다. 보다 효율적으로 구성하는 방식인데 이렇게 하면 button이 눌린 순간 모든 main 프로그램이 멈추고 interrupt를 수행한 후 다시 프로그램이 동작된다. Interrupt가 최우선 과제가 되기 때문에 보다 정확하고 깔끔한 동작을 할 수 있다. 코드도 훨씬 간결하다. 무엇보다 interrupt를 수행하는 이유는 button을 누르는 동작에 대해 정확하게 반응한다는 것에 있다. Button을 무작정 길게 누른다거나 아주 빠르게 누르고 떼서 미처 반복문에서 이를 인지하지 못한다고 하더라도 interrupt를 사용하면 설정한 트리거에 맞게 정확히 반응한다. 이러한 이점으로 button제어의 큰 algorithm을 interrupt를 이용하여 코드를 작성하려고 했으나 알 수 없는 오류로 계속해서 동작이 제대로 되지 않았다. 따라서 어쩔 수 없이 switch-case구문을 이용하여 코드를 작성했다.  먼저 direction control의 경우 여자 방식을 반대로 해주어야 해서 단계가 꽤 복잡하여 별도의 함수로 구현했다. 더하여, direction을 나타내는 변수 direction이 시계방향인지(CW) 반시계 방향인지(CCW)에 따라 그 값을 바꿔주는 함수도 따로 만들었다. 이 함수들은 이후 3번째 button이 눌릴 때 호출된다.  또한 속도를 제어하기 위해 delay를 주어 control했는데, delay가 작을수록 빠르게 도는 것처럼 보이고, delay가 크면 느리게 motor가 회전하는 것으로 보이기 때문에 각 버튼이 눌릴 때 delay를 조절하여 속도를 간접적으로 control하였다.  각 핀과 변수들이 초기화 되고 나서, main함수 안에서 switch-case문이 나타난다. while반복문 안에 위치하기 때문에 계속해서 입력을 검사하고 동작한다. 각 case는 1,2,3button입력에 대해 동작하고 아무것도 없는 경우 default로 처리했다.  이후 button으로 속도를 늘리고 줄이는 것이 아니라 switch를 이용하여 절대 속도를 설정한 동작도 해보았는데 역시 동일하게 동작했다.    그림 2. 실험 2 회로 구성도   1. **실험 결과**    1. 실습 1단계     그림 4. 실험 1 실행 결과1    그림 5. 실험 1 실행 결과2   * 1. 실습 2단계     그림 6. 실험 2 실행 직후    그림 7. 실험 2 실행 결과    그림 8. 실험 2 실행 결과   1. **고찰**   이번 주차에는 stepping motor로 실험을 진행했다. 이제까지 Servo Motor, DC Motor, 그리고 이번 주 Stepping Motor까지 세 종류의 Motor에 대해서 공부하고 실험해 보았다. 3주에 걸쳐 연속적으로 motor를 공부해보니 차이점이 한 눈에 들어와 이를 간단히 정리해볼까 한다.  먼저 Servo Motor는 가장 값이 싸다고 알고 있고 간단하다. 특정 각도로 움직이는 데에 많이 쓰이는 것 같다. 예를 들면 거중기나 엘리베이터 등 어떤 각도로만 움직이는 동작을 할 때 쓰이는 것을 알 수 있었다.  다음으로 DC Motor는 직류 전류로 동작하는 모터로, 선풍기나 청소기 등에 쓰이는 모터가 이 DC Motor라고 한다. 단순히 빠르고 강력하게 회전이 필요하다면 이 모터를 쓰면 될 것 같다.  마지막으로 이번주에 배운 stepping motor는 직접 값을 써서 코일에 자기장을 인가하여 보다 정밀한 각도로 동작하게 할 수 있다. 물론 한 방향으로 도는 것도 가능하지만 주로 로봇 팔 등 정확한 동작을 위해 사용하는 모터라고 한다.  이번 주 실험 자체는 어렵지 않았지만 motor에 대해 자세히 알게 되었다. 기회가 있다면 다른 여자방식도 이용하여 보다 정밀한 동작을 구동하고 싶다. | | | | |
|  | | | | |