|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **예 비 보 고 서** | | | | |
| 학 과 | 학 년 | 학 번 | 조 | 성 명 |
| 전자공학과 | 3 | 12191505 |  | 윤수연 |
| 실험 제목 | ADC/Acceleration Sensor | | | |
| 1. **자료조사**    1. **ADC란**   ADC란 Analog-to-digital converter의 약자이다. 즉 연속적인 analog 신호를 0과 1로 discrete하게 구성된 digital 신호로 변환하는 소자다. 반대로 디지털신호를 아날로그신호로 변환해 주는 소자는 DAC라고 한다. 아날로그 신호는 프로세서의 외부에 있는 전기적 신호나 물리적 신호를 수학적으로 2진수가 아닌 실수의 범위 내에서 변하는 신호를 말하고 이는 수학적으로 표현하기 어려운 신호를 프로세서가 인식할 수 있는 정수의 범위로 일정 시간마다 하나의 데이터로 변환하는 과정이 필요한데 이를 ADC가 수행하는 것이다. 그래서 각종 센서를 MCU가 인식하게 하는 데에 필수적이다. 예를 들어 LED의 밝기를 1비트로 표현한다면 1은 밝은 상태, 0은 어두운 상태로 표현할 수 있다.       * 1. **ADC의 성능**   ADC의 성능은 크게 두 가지로 측정된다. 첫번째 분해능, 두번째 sampling rate다. 두 능력이 높을수록 원래 아날로그 신호와 유사한 신호를 표현할 수 있고 즉 원래 신호와 비교해서 오차가 없다는 뜻이므로 성능이 좋다고 할 수 있다.   1. 분해능   분해능은 아래 그림에서 눈금의 미세함을 나타낸다. 필요한 분해능은 용도와 기기에 따라 다르다.    ADC에 대해서 알아보자 :: OSHW Alchemist   1. Sampling rate   디지털 신호는 시간방향에 대해서 분산적이기 때문에 샘플링이 필요한데 이 샘플링의 주기에 따라 ADC의 성능이 달라진다. 이 또한 ADC의 용도에 따라 다르다. 아래 그림에서 샘플링 속도가 높을 수록 더욱더 촘촘하게 값을 측정한 것을 알 수 있다. 촘촘하게 측정할수록 아날로그 값을 왜곡 없이 디지털화 시킬 수 있기 때문이다. 따라서 sampling rate가 높으면 시간측 분해능이 좋아진다고 말할 수 있다.  Video ADC｜자인일렉트로닉스 - THine  분해능과 sampling rate는 용도마다 다른데, 그 값을 아래에 표시했다.     * 1. **ATmega128의 ADC**   아트메가128에 내장된 ADC는 10비트 분해능에 8개의 채널이 있다. 10비트 분해능이니 1024개의 단계로 나뉘어 진다는 것을 알 수 있다. 입 관련 핀은 아래의 그림과 같다. ADC 7~0까지 아날로그 입력 핀, AVCC로 ADC전원핀, AREF로 참조 전압(Reference Voltage)를 받는 핀으로 세 종류다. 즉 총 10개의 핀이 된다.  ATmega128 강좌 - ADC - 네로라르크님께 바칩니닸 : 네이버 블로그  8개의 채널은 Analog신호를 받을 수 있는 핀이 8개라는 뜻이다. 그리고 AVCC라는 핀에 추가 전압을 넣어줘야 전원이 켜지고 ADC를 동작 시킬 수 있다. 입력 값의 최저 값을 GND (0V)로, 최대값을 AREF로 사용하며, 사용자 설정에 따라 AVCC 핀을 활용 가능하다. 아날로그 신호 하나를 디지털화 하는 것에는 약 100us가 소요된다. 그런데 analog 신호를 받는 입력방식은 두 가지다.   1. 단극 입력: single ended input. 기준전압이 되는 Vref에 대한 비율을 변환한다. 이 때 Vref는 AREF pin에서 들어온 입력을 말한다. 2. 차동 입력: difference input. 단자 간 전압 차를 변환한다. 두 핀이 필요하다. 3. 증폭률: gain. 입력 전압이 너무 약할 경우 ADC 안에서 신호를 증폭하여 변환하는데, 이때의 증폭률을 말한다.   그리고 Multiplexer를 통해서 총 8개의 핀과 ADC 설정 핀을 선택적으로 사용할 수 있으며, ADC 값을 증폭시켜주는 amplifier를 선택 가능하다. 이 동작은 레지스터의 설정으로 가능하다. ADC의 샘플링 주기는 프로세서 클럭에 영향을 받고, 이를 조절하기 위한 프리스케일러가 회로 내부에 존재한다. 프리스케일러는 초 당 샘플링 속도를 제어하며 시스템 내부 클럭을 1/n으로 쪼개 샘플링 펄스로 사용한다. 이 때 n의 default factor는 2다.  또한 아트메가128에서 ADC는 두가지 모드가 있다. 단일 변환 모드(SIngle Conversion Mode)라고 한 번씩 변환하는 모드와 연속 변환 모드(자유 동작 모드 free running mode)라고 연속해서 변환하는 모드가 있다. 단일 변환 모드의 경우 ADSCRA레지스터의 ADSC=1(ADC Start Conversion)로 설정함으로써 시작되고 변환이 끝나면 자동으로 0으로 클리어 된다. 마치 인터럽트처럼 변환이 발생한다. Conversion이 진행 중일 때는 1값으로 유지되며 동시 동작을 거부한다. 그래서 만약 AD변환중 입력채널이 바뀌었다면 ADC는 현재의 변환을 마치고 새로 선택된 채널로 변경된다.  다음으로 연속 변환 모드(자유 동작 모드 free running mode)는 연속해서 변환하는 모드로 ADSCRA레지스터의 ADFR=1로 설정함으로써 ADC는 지속적으로 샘플링과 변환을 수행하여 ADC데이터 레지스터를 갱신할 수 있다. 데이터를 계속해서 업데이트 하는 것이다. 참고로 converter는 하나이기 때문에 Mux를 통해서 데이터 입력 핀을 원할 때 바꿀 수 있지만 이러한 동작들을 고려하여 conversion중에는 핀을 변경하지 않도록 해야 한다. ADC관련 레지스터는 ADMUX,ADCSRA,ADCH/L이 있다.   * 1. **ADC제어 레지스터**      1. ADMUX(ADC Multiplexer Selection Register)     :   * Bit 7,6: REFS1,REFS0(Reference Selection Bit): ADC에서 사용하는 기준전압을 정한다. * Bit 5: ADLAR(ADC Left ADjust Result). 변환 결과가 ADC레지스터에 저장될때 정렬방식을 정하는데 사용한다. 1은 변환 결과를 ADCH/L에 저장할 때 좌측으로 끝을 맞추어 저장, 0은 변환 결과를 ADCH/L에 저장할 때 우측으로 끝을 맞추어 저장. * Bit 4,3,2,1,0: MUX4,MUX3,MUX2,MUX1,MUX0 (Analog Channel and Gain Selection Bit). ADC변환기의 아날로그 입력 채널 및 gain을 결정        * + 1. ADCSRA(ADC Control and Status Register A)     :   * Bit 7: ADEN(ADC Enable). 1: ADC활성화, 0: ADC비활성화 * Bit 6: ADSC(ADC Start Conversion). 1은 ADC변환 시작. (단일 변환모드일 때 1번만 작동, free running 모드일 때 변환 동작 반복) 0은 ADC변환이 시작되지 않음. * Bit 5: ADFR (ADC Free Running Select). 1: 연속 변환 모드, 0: 단일 변환 모드 * Bit 4: ADIF(ADC Interrupt Flag). ADC 변환이 되면 1로 set되고 변환완료 인터럽트를 요청한다. 이때 SREG의 I비트가 1이고 ADIE비트가 1이면 인터럽트가 발생되어 처리된다. 인터럽트가 처리되면 ADIF가 0으로 클리어 된다. 물론 1을 넣어도 클리어 된다. * Bit 3: ADIE(ADC Interrupt Enable). 1은 ADC Interrupt enable. (이때 SREG레지스터의 I비트가 SET되어있어야함) 0은 ADC Interrupt disable. * Bit 2,1,0: ADPS 2~0 (ADC prescaler Select Bit). ADC에 인가되는 클럭의 분주비를 설정한다.      * + 1. ADCH/L (ADC Data Register)     ADC의 변환결과를 저장한다. ADMUX레지스터의 ADLAR에 따라 변환결과를 저장하는 방식이 다르기 때문에, ADC변환이 되었을 때 ADCH , ADCL 두 레지스터에 값이 저장된다. 왜냐하면 ATmega128에서 사용하는 ADC는 10비트 ADC이기 때문에 레지스터의 값들이 두 곳에 저장된다. ADLAR = 0일때 ADCL은 0~255이고 ADCH은 0~768이라서 총 값이 0~1023으로 1024개로 표현할 수 있는 걸 볼 수 있다.   * 1. **Acceleration Sensor**   동작인식 센서의 하나인 가속도 센서는 단위 시간당 직선운동에 대한 속도의 변화를 측정하는 센서이다. 우리가 알고 있는 그 가속도(Acceleration)를 측정하는 센서로, 속도(Velocity)가 아닌, 가속도(Acceleration)를 측정한다는 것에 주의해야한다. 예를 들어 여러 가속도 중에서 중력가속도(Gravity)는 우리가 항상 느끼고 있는 가속도인데, 이에 가속도 센서는 기본적으로 중력 방향에 있을 때 중력 가속도 9.8 (m/s^2)을 표시한다. 이는 스마트폰이나 태블릿 화면을 가로로 표시할지 세로로 표시할지를 결정하는 것처럼 중력이 작용하는 방향을 탐지하여 디바이스 사용자의 편의를 돕는 데에 사용할 수 있다. 당연히 일반적 상황에서 가속도 값은 analog 신호다.  가속도 센서로 측정되는 가속도 값은 일반적으로 동적 가속도(dynamic acceleration) 값과 정적 가속도(static acceleration) 값을 함께 사용한다. 동적 가속도는 진동이나 충격 등에 의해 발생하는 가속도를 말하며, 정적 가속도는 중력 가속도를 말한다. 선형 가속도 센서는 가속도 센서가 생성하는 값들 중에서 중력 가속도를 제외한 값으로 3차원 벡터 값을 제공하며, 가속도 센서가 생성한 값을 가공해서 만들어지므로 소프트웨어적인 센서에 해당한다.  가속도를 측정하는 원리는 아래 두 공식을 보면 알 수 있다.  (1) 뉴턴의 제 2법칙 (가속도의 법칙)  F = MA  (2) 휴크 (Hooke)의 법칙 (용수철 법칙)  F =kX  여기서 여기서 F를 제거하면 MA = kX 다. 이는  A = (kX) /M  로 정의되고 즉, k (스프링 상수) 가 일정하고 M (스프링에 달린 물체의 질량) 이 일정할 경우 가속도는 스프링이 늘어난 길이에 정비례함을 나타낸다. 따라서 아래 그림과 같이 가속도를 측정할 수 있다.  가속도 센서와 자이로 센서 : 네이버 블로그  3축 가속도 센서모듈 ver.3 / 디바이스마트   1. **실험**    1. **실험 예상**       1. **실습 1단계**   가변저항을 연결하여 PF0에 다양한 전압 값이 들어오는 상황에서 ADC를 통하여 실시간으로 측정하여 전압 값을 LCD로 출력    제공되는 1단계 코드를 보니 다소 복잡하다. 하지만 LCD동작을 위한 부분은 익숙하기 때문에 제외하면 아날로그 값을 read하는 부분 빼고는 크게 어려운 부분이 없다. 그런데 프로테우스는 가상 하드웨어 프로그램인데 디지털로 아날로그 신호를 구현하는 것이라면 꽤 높은 분주비나 sampling rate가 필요할 것이다. 다양한 전압 값이 어떻게 들어올지는 모르겠지만 아날로그 신호만 제대로 들어온다면 실험을 성공할 수 있을 것 같다.   * + 1. **실습 2단계**   Acceleration sensor를 연결하여 가속도 값이 실시간으로 들어오 는 상황에서 ADC를 통하여 1초마다 X축/Y축/Z축 값을 USART로 출력  • 아래 그림은 Proteus에 가속도 센서가 없기 때문에 가변 저항 3개를 대신 연결해 놓음    연속적으로 값이 갱신되는데 USART로 통신해야 하는 과정이다. USART는 항상 어려웠기도 하고 계속해서 값을 갱신하는 것이 가능할지 모르겠다. 특히나 가상 포트를 이용한다고 한들 실제 센서가 없는데 아날로그 신호처럼 신호를 받을 수 있을까?   * + 1. **실습 3단계**   • ADC/Acceleration sensor를 활용한 새로운 시나리오 제안 및 구현  • 2단계 이상의 난이도를 제안해야 가산점 있음  ADC는 모든 센서의 기본이고 Acceleration sensor도 워낙 많이 쓰일 수 있는 소자라 새로운 시나리오를 짜는 것이 편리할 것 같다. 지난 시간에 사용한 stepping 모터와 함께 acceleration sensor의 값에 따라 회전하는 기기를 만들어도 좋을 것 같다. | | | | |
|  | | | | |