0. Introduction

1) Encoder

요약) 우리가 사용한 엔코더 특징을 데이터시트를 바탕으로 요약

엔코더는 물리적(아날로그) 신호를 디지털 신호로 변환하는 센서이다. 엔코더는 센서와 트렌스듀서로 구성되어, 회로의 스위치로서 작동하여 기계적인 위치의 변화나 방향 또는 각도 등을 검출할 수 있다. 스마트 센서와 IoT디바이스에서 사용하는 엔코더는 BOURNS사의 PEC-16 (Incremental encoder)로 그림 1과 같은 도면을 갖고 있으며 각도에 따라 스위치를 On(closed switch일 때 최대 ) 또는 Off(Open처럼 작동할 때 ) 형식으로 작동한다. 동작할 때 최대 RPM은 100이므로 이 이상으로 엔코더를 회전할 시에 정확한 신호를 받지 못할 수 있다. 엔코더에는 A채널과 B채널이 있고 시계 방향으로 돌린다면 A채널의 Pulse가 먼저 관측되며 이어서 B채널의 Pulse가 뒤따라 관찰된다. 반대로 반시계 방향으로 엔코더를 돌릴 경우에 B채널의 파형이 먼저 나오고 이어 A채널에서 Pulse가 나오는 것을 확인할 수 있다. 이 때 두 신호의 위상차는 90도임은 output이 2bit quadrature code임에서 알 수 있다. 또한 이를 통하여 본래 A채널만 사용하는 것을 기준으로 1 pulse (1 사각파) 당 15도씩 각도 차이가 발생하는 것에서 2채널의 falling edge와 rising edge를 사용함을 통해 3.75도로 해상도를 높일 수 있다는 장점이 있다. Contact rating은 1mA이며 정격 전류를 흘려주기 위하여 회로에 충분한 저항 값을 설치해야 한다.

도표이(가) 표시된 사진

자동 생성된 설명 텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

그림 그림

1. Rotational angle in CW using A signal

1) Make a circuit to measure A signal from channel A of the encoder

|  |  |
| --- | --- |
| 회로사진 | 설명 |
|  | 저항 (-)단 & DAQ로 전압 추출  Power supply 양극은 저항 (+)단에 연결되어 있다.  만약 Encoder의 각도가 기준각에서 15도의 정수배만큼 바뀌게 된다면, 저항에서 전압 강하가 일어나지 않고 저항의 (-)단이 위치한 노드에서 전압강하가 5V 검출된다. |

2) Write a Matlab code that reads the voltage of A signal and find resolution (pulse per rotation, PPR)  
- Sample rate: 100 hz  
- Duration time: 10 s

|  |  |
| --- | --- |
| 코드 | 설명 + 주석 |
| k=0;  totalrotation= zeros(length(time\_stack),1);  for i=2:length(time\_stack)  if data\_stack(i-1,1)< 1.5 && data\_stack(i,1)>1.5  k=k+1;  totalrotation(i,1)= totalrotation(i-1,1)+7.5;    elseif data\_stack(i-1,1)>1.5 && data\_stack(i,1)<1.5  k=k+1;  totalrotation(i,1)= totalrotation(i-1,1)+7.5;    else  totalrotation(i,1) = totalrotation(i-1,1);  k=k;    end  end  figure(1); plot(time\_stack,data\_stack,'r-');  title('Pulse of encoder in time');  xlabel('Time [sec]');  ylabel('Pulse [V]');  figure(2); plot(time\_stack,totalrotation,'b-');  title('Angle derived from encoder pulse');  xlabel('Time [sec]');  ylabel('Angle [deg]'); | Total rotation: 총 회전한 각도, 현재 k의 7.5배  반복문 안에서의 조건문:   1. 전 인덱스의 data\_stack value가 threshold value(1.5)보다 작고 현 인덱스 값이 threshold value보다 클 때, 변화가 일어난 것을 감지하여 k를 업데이트한다. 2. 전 인덱스의 data\_stack value가 threshold value(1.5)보다 크고 현 인덱스 값이 threshold value보다 작을 때, 변화가 일어난 것을 감지하여 k를 업데이트한다. 3. 그 외의 경우 threshold보다 두 인덱스의 값이 동시에 작거나 동시에 클 때는 k를 업데이트 하지 않는다. |
| 결과 | 설명 |
|  | 첫번째 그래프는 회로의 엔코더 각도에 따른 Pulse이다.  천천히 회전할 시 5V의 전압이 측정되나 빠른 속도로 회전을 시킬 경우에 더 적은 전압 값이 나올 수 있으므로 Threshold value는 1.5V로 정했다. 기준 시간(10초)동안 Pulse의 변화를 관찰할 수 있다.  두번째 그래프는 엔코더의 전압에 따라 도출된 각도를 도시한다. Pulse 한번 당 15도의 각도 변화를 관찰하였다. 엔코더의 전압과 각도에 대해 관찰했을 때 Rising edge에서 7.5도 Falling edge에서 7.5도 총 15도가 변화하는 것을 확인했다.  마지막 그래프에서 볼 수 있듯이 엔코더의 A채널 회로로 도출한 각도 해상도는 7.5도이다. |

3) Modify your code to display the current rotational angle in clockwise  
- Read rising edge of A signal  
- Range: 0~360 deg

|  |  |
| --- | --- |
| 코드 | 설명 + 주석 |
| %=============================================1-3)=============================================%  k= zeros(length(event.Data),1);  if length(data\_stack) == length(event.Data)  for i= 2:length(event.Data)  if event.Data(i-1,1)<=1.5 && 1.5 < event.Data(i,1)  k(i,1)= k(i-1,1) + 1; %시계방향  else  k(i,1)= k(i-1,1); %아닐 시 k는 그 전값과 동일한 값을 가짐  end  end  angle\_stack= k \* 15; %k에 15배(각도)  else  if data\_stack(length(data\_stack)-length(event.Data),1) <=1.5 && 1.5 < event.Data(1,1)  k(1,1) = 1; %시계방향 마지막 data\_stack 값과 event.Data의 첫 값을 비교  end  for i= 2:length(event.Data)  if event.Data(i-1,1)<=1.5 && 1.5 < event.Data(i,1)  k(i,1)= k(i-1,1) + 1; %시계방향  else  k(i,1)= k(i-1,1); %현위치  end  end  angle\_buffer = ones(length(event.Data),1) \* angle\_stack(length(angle\_stack),1); %angle\_stack의 마지막 값이 들어간 event.Data크기만큼의 배열  angle\_stack = [angle\_stack; angle\_buffer + k \* 15]; %angle\_stack 업데이트  end  plot(time\_stack,angle\_stack); hold on;grid on;  plot(time\_stack, data\_stack);  title('Angle derived from Pulse');  xlabel('Time [sec]');  ylabel('Angle in real time [degree] and A channel signal [V]'); | 전 값과 비교하여  만약 Rising edge가 검출될 시에 각도를 15도 업데이트 하는 알고리즘  Rising edge 실시간으로 검출하는 법:   * 전 값과 비교하였을 때 전 값이 넘지 못하는 threshold value(1.5V)를 현 값이 넘었을 때 현 k값을 전 값에 대해 업데이트 (+1)   Angle\_stack에 업데이트하는 법:   * 마지막 angle\_stack의 값으로 Buffer 배열을 만든 뒤, k에 해상도를 곱하여 더함. |
| 결과 | 설명 |
|  | A 채널의 신호를 받는 동시에 각도가 업데이트 되는 것을 관찰할 수 있음.  실시간 plotting에서 resolution은 15도로 관측 되었다. |

2. Angular position & direction – A, B signals

1) Make a circuit to measure A and B signals from the encoder

|  |  |
| --- | --- |
| 회로사진 | 설명 |
| 텍스트이(가) 표시된 사진  자동 생성된 설명  전자제품이(가) 표시된 사진  자동 생성된 설명 | 도표, 개략도이(가) 표시된 사진  자동 생성된 설명  그림 3 엔코더를 스위치로 대체한 등가회로  만약 Channel A의 Switch가 Open이라면 전압원의 전체 전압(5V)이 A0에 걸린다.  만약 Channel A의 Switch가 Closed라면  Channel B가 Closed면  Channel B가 Open이면  스위치가 Encoder이며 특정각에서 Close 또는 Open하는 특성을 갖는다. |

2) Write a Matlab code that reads the voltage of A and B signals  
- Sample rate: 100 hz  
- Duration time: 10 s

|  |  |
| --- | --- |
| 코드 | 설명 + 주석 |
| ch(1) = addAnalogInputChannel(mydaq,'Dev2',0,'Voltage'); %mydaq, device number, pin number  ch(1).Range = [-10.0 10.0];  ch(1).TerminalConfig = 'SingleEnded';  ch(2) =addAnalogInputChannel(mydaq,'Dev2',1,'Voltage'); %mydaq, device number, pin number  ch(2).Range = [-10.0 10.0];  ch(2).TerminalConfig = 'SingleEnded';  lh = addlistener(mydaq,'DataAvailable', @listener\_callback\_week4\_p2);  startBackground(mydaq);  %%  plot(time\_stack,data\_stack(:,1)); hold on; plot(time\_stack,data\_stack(:,2));  title('A and B channel signals of encoder');  xlabel('Time [sec]');  ylabel('Voltage [V]');  legend('channel A', 'channel B');  ylim([-0.5 5.5]); | 두가지의 채널을 받아올 수 있도록 채널 2를 만들었다.  Data\_stack이 두 채널의 신호를 동시에 받아오고, 첫번째 열은 A채널, 두번째 열은 B채널이기 때문에, 따로 구분하여 plotting 하였다. |
| 결과 | 설명 |
|  | 채널 A 시그널과 채널 B 시그널 파형이 번갈아가며 앞서기 때문에 엔코더는 번갈아 가며 왕복 운동하는 것을 유추할 수 있다.  약 3.5초까지는 활발한 신호의 변화로 엔코더의 회전이 있다고 추측하지만 그 이후로 신호의 변화가 뚜렷하게 바뀌지 않기 때문에 각도의 변화가 없다고 예상한다. |

3) Modify your code to display the current rotational angle  
- Read the rising edges of A signal for angular position  
- Refer to the state of B signal for rotational direction  
- Range: 0~360 deg, Sign : CCW(+), CW(-)

|  |  |
| --- | --- |
| 코드 | 설명 + 주석 |
| %=============================================2-3)=============================================%  k= zeros(length(event.Data),1);  if length(data\_stack) == length(event.Data)  for i= 2:length(event.Data)  if event.Data(i,2) <= 1.5 && event.Data(i-1,1)<=1.5 && 1.5 < event.Data(i,1)  k(i,1)= k(i-1,1) - 1; %시계방향  elseif 1.5 < event.Data(i,2) && event.Data(i-1,1)<=1.5 && 1.5 < event.Data(i,1)  k(i,1)= k(i-1,1) + 1; %시계방향  else  k(i,1)= k(i-1,1); %아닐 시 k는 그 전값과 동일한 값을 가짐  end  end  angle\_stack= k \* 15; %k에 15배(각도)  else  if event.Data(1,2) <= 1.5 && data\_stack(length(data\_stack)-length(event.Data),1) <=1.5 && 1.5 < event.Data(1,1)  k(1,1) = -1; %시계방향 마지막 data\_stack 값과 event.Data의 첫 값을 비교  elseif (1.5 < event.Data(1,2)) && (data\_stack(length(data\_stack)-length(event.Data),1) <=1.5) && (1.5 < event.Data(1,1))  k(1,1) = 1; %반시계방향 마지막 data\_stack 값과 event.Data의 첫 값을 비교  end  for i= 2:length(event.Data)  if event.Data(i,2) <= 1.5 && event.Data(i-1,1)<=1.5 && 1.5 < event.Data(i,1)  k(i,1)= k(i-1,1) - 1; %시계방향  elseif 1.5 < event.Data(i,2) && event.Data(i-1,1)<=1.5 && 1.5 < event.Data(i,1)  k(i,1)= k(i-1,1) + 1; %반시계방향  else  k(i,1)= k(i-1,1); %현위치  end  end  angle\_buffer = ones(length(event.Data),1) \* angle\_stack(length(angle\_stack),1); %angle\_stack의 마지막 값이 들어간 event.Data크기만큼의 배열  angle\_stack = [angle\_stack; angle\_buffer + k \* 15]; %angle\_stack 업데이트  end  plot(time\_stack,angle\_stack);  %drawnow;  title('Angle derived from Pulse');  xlabel('Time [sec]');  ylabel('Angle in real time [degree]');  end | 배열 k는 eventData의 배열의 크기를 갖는다.  eventData의 B채널(On/Off)을 기준으로 A채널의 Rising edge에 따라 +방향 또는 -방향인지를 현재 배열 값에 업데이트 해준다. 이때 flag는 1 또는 -1를 이전 k배열값에 더하여 준다.  Angle stack에 차곡히 k값에 15배하여 마지막 값에 대입해 업데이트 하여 준다. |
| 결과 | 설명 |
|  | Resolution: 15degrees  시계 방향🡪반시계 방향으로 회전이 관찰 됌  약 3초 동안 회전 후 각도 유지 |

4) Double the resolution by utilizing rising and falling edges of A signal

|  |  |
| --- | --- |
| 코드 | 설명 + 주석 |
| %=============================================2-4)=============================================%  k = 0;  angle = zeros(length(time\_stack),1);  for i=2:length(data\_stack)  if data\_stack(i,2) <= 1.5 && data\_stack(i-1,1) <= 1.5 && data\_stack(i,1) > 1.5  k = k - 1;  angle(i,1) = k \* 7.5;  elseif data\_stack(i,2) > 1.5 && data\_stack(i-1,1) > 1.5 && data\_stack(i,1) <= 1.5  k = k - 1;  angle(i,1) = k \* 7.5;    elseif data\_stack(i,2) > 1.5 && data\_stack(i-1,1) <= 1.5 && data\_stack(i,1) > 1.5  k = k + 1;  angle(i,1) = k \* 7.5;  elseif data\_stack(i,2) <= 1.5 && data\_stack(i-1,1) > 1.5 && data\_stack(i,1) <= 1.5  k = k + 1;  angle(i,1) = k \* 7.5;  else  angle(i,1) = k \* 7.5;  end  end  figure(2);  plot(time\_stack,angle);  title('Encoder signal to Angle');  xlabel('Time [sec]');  ylabel('Angle [degree]');  xlim([0 10]); | K는 초기 각도를 0으로 해주기 위해 0으로 설정을 하였다.  만약 채널 B이 Off이지만 A가 On이 된 시점에 7는 시계방향으로 돈 것이기 때문에 시계방향(-)으로 7.5 업데이트 하여준다.  채널 B가 On일 때 A가 Off가 되면 이 또한 시계방향임을 인지하고 시계방향으로 업데이트 한다.  채널 B가 On인데 A가 On이 된다면 이는 반시계 방향이다. 따라서 기준 각도에서 7.5를 더해 준다.  채널 B가 Off인데 A가 Off가 된다면 반시계방향이므로 7.5도를 더해 준다.  파형이 항상 겹치기 때문에 B채널을 기준으로 A의 업데이트에 따른 각도 명령을 주었다. |
| 결과 | 설명 |
|  | 시계 방향으로 돌린 후 반시계 방향으로 돌린 형태이며 실제로 엔코더를 시계 방향으로 돌린 후 반시계 방향으로 돌렸다.  이 때 채널 A와 B에 들어온 신호를 같이 비교해 보면, 초반시점에는 (시계 방향으로 돌렸을 때), A채널에서 신호가 먼저 들어오지만, 반시계 방향으로 돌린 경우에 B채널에서 신호가 들어오는 것을 확인할 수 있었다. 또한 A채널 B채널 신호가 항상 곂치고 다음 신호에 간섭 받지 않기 때문에 이러한 알고리즘을 짜게 되었다. |

5) Quadruple the resolution by utilizing rising and falling edges of both A and B signals

|  |  |
| --- | --- |
| 코드 | 설명 + 주석 |
| %=============================================2-5)=============================================%  k=0; % Number of  angle = zeros(length(time\_stack),1);  Threshold= 2; %Threshold voltage  for i=2:length(data\_stack)  if (data\_stack(i,1) <= Threshold) && (Threshold < data\_stack(i,2)) && (data\_stack(i-1,2) <= Threshold)  k = k + 1; %A는 OFF상태, B는 TURNING ON => 반시계방향  angle(i,1)= k \* 3.75;  elseif (Threshold < data\_stack(i,2)) && (data\_stack(i-1,1) <= Threshold) && (Threshold < data\_stack(i,1))  k = k + 1; %B는 ON상태, A가 TURNING ON => 반시계방향  angle(i,1)= k\*3.75;  elseif (Threshold < data\_stack(i,1)) && (Threshold < data\_stack(i-1,2)) && (data\_stack(i,2) <= Threshold)  k = k+1; %A는 ON상태, B가 TURNING OFF => 반시계방향  angle(i,1)= k\*3.75;  elseif (data\_stack(i,2) <= Threshold) && (Threshold < data\_stack(i-1,1)) && (data\_stack(i,1) <= Threshold)  k = k+1; %B는 OFF상태, A가 TURNING OFF => 반시계방향  angle(i,1)= k \* 3.75;    elseif (data\_stack(i,2)) <= Threshold && (Threshold < data\_stack(i,1)) && (data\_stack(i-1,1) <= Threshold)  k = k - 1; %B는 OFF상태, A는 TURNING ON => 시계방향  angle(i,1)= k\*3.75;  elseif (Threshold < data\_stack(i,1)) && (data\_stack(i-1,2) <= Threshold) && (Threshold < data\_stack(i,2))  k = k - 1; %A는 ON상태, B가 TURNING ON => 시계방향  angle(i,1)= k\*3.75;  elseif (Threshold < data\_stack(i,2)) && (Threshold < data\_stack(i-1,1)) && (data\_stack(i,1) <= Threshold)  k = k - 1; %B는 ON상태, A가 TURNING OFF => 시계방향  angle(i,1)= k\*3.75;  elseif (data\_stack(i,1) <= Threshold) && (Threshold < data\_stack(i-1,2)) && (data\_stack(i,2) <= Threshold)  k = k - 1; %A는 OFF상태, B가 TURNING OFF => 시계방향  angle(i,1)= k\*3.75;  else  angle(i,1)= k\*3.75;  end  end  figure(3);  plot(time\_stack,angle); hold on; grid on;  plot(time\_stack,data\_stack(:,1),'r'); plot(time\_stack,data\_stack(:,2),'g');  legend('Angle', 'Channel A signal', 'Channel B signal');  title('Encoder signal to Angle [Resolution: 3.75]');  xlabel('Time [sec]');  ylabel('Angle [degree]');  xlim([0 10]); | 반시계 방향일 때:   * A는 OFF상태, B는 TURNING ON 🡺 반시계 방향 * B는 ON상태, A가 TURNING ON 🡺 반시계 방향 * A는 ON상태, B가 TURNING OFF 🡺 반시계 방향 * B는 OFF상태, A가 TURNING OFF 🡺 반시계 방향   시계방향일 때:   * B는 OFF상태, A는 TURNING ON 🡺 시계 방향 * A는 ON상태, B는 TURNING ON 🡺 시계 방향 * B는 ON상태, A는 TURNING OFF 🡺 시계 방향 * A는 OFF상태, B가 TURNING OFF 🡺 시계 방향   상수 k는 매 조건 마다 + 혹은 -하거나 어떤 조건도 만족하지 않을 시에는 업데이트하지 않음.  매 순간마다 업데이트 되는 k에 resolution을 곱한 것이 기준각도(0도)에서의 각도 차 |
| 결과 | 설명 |
|  | 바로 이전의 그래프와 비교하였을 때 확실히 Continuous signal과 비슷한 형태를 띔. |

6) Is higher resolution always a better choice? Why or why not? Discuss in terms of actuator’s rotational speed and sampling frequency

신호의 파형만 놓고 보았을 때, 확실히 7.5의 해상도를 갖는 그래프와 3.75의 그래프는 극명한 차이를 지닌다. 3.75 해상도를 가질 때, 더 Continuous에 가까운 곡선으로 보인다.

다만, 시간당 처리해야 할 연산량이 많아지게 되고 받아야 할 데이터 개수가 많아지게 되어 실시간 반영에 대해 delay가 일어나거나 오류 또는 과부하가 날 확률이 높아진다.

또한 Resolution이 좋은 센서는 그렇지 않은 센서보다 가격이 높다. 적절한 해상도에 타협하는 것이 바람직하다.