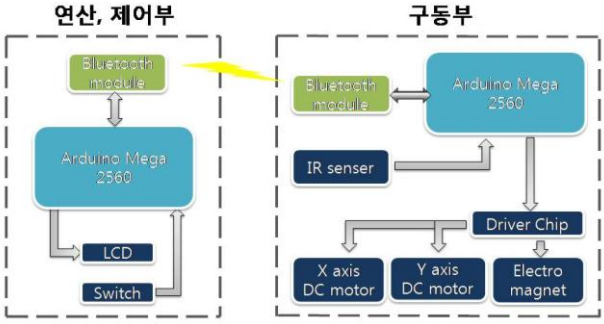
3. 블루투스 기반 체스 로봇 개발에 관한 연구

3.1 하드웨어 구현

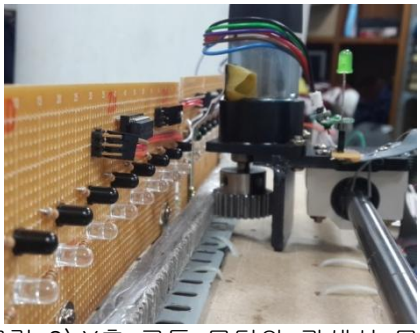
체스를 플레이 할 수 있도록 만들어진 본 장치(이하 '체스 로봇')는 크게 두 부분으로 나누어진다. 사용자의 입력을 받음과 동시에 체스프로그램을 연산하는 제어부, 그리고 모터와 전자석을 이용해 실질적으로 말을 이동시키는 구동부로 구성되어있다. 체스 로봇의 전체 구성도는 (그림 1)과 같다. 제어부와 구동부 모두 개발의 편의를 위해 마이크로 컨트롤러로 아두이노를 사용하고 있다. 제어부와 구동부는 블루투스를 통해 연결되어있으며 이를 통해 이동시킬 말의 좌표정보를 제어부에서 구동부로 전달한다. 제어부(그림 2)는 체스프로그램이 담겨저있는 아두이노와 좌표입력을 위한 스위치, 그리고 상태확인을 위한 LCD로 구성되어있다. 구동부의 경우 모터 제어을 위한 코드가 담긴 아두이노와 모터의 위치를 감지하는 광센서 모듈, 구동 모터와 말을 잡아줄 전자석을 포함하고있다. 체스 말 하단에는 자석이 부착되어있어 전자석을 통해 이동이 가능하도록 하였다.



(그림 1) 체스로봇의 전체 구성도

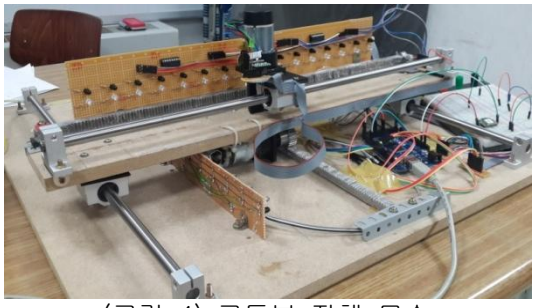
(그림 2) 제어부의 모습

구동에는 기어드 DC모터를 사용하였으며 모터의 위치 측정을 위해 16조의 광센서를 갖는 모듈을 제작하여 각 축당 16개의 위치 포인트를 감지할 수 있도록 하였다(그림 3).



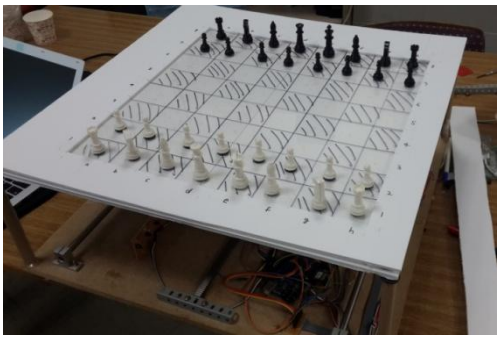
(그림 3) Y축 구동 모터와 광센서 모듈

X축 구동부 위에 Y축 구동부거 올러가는 형태이며 전체적인 구성의 내부는 그림. 4를 참조.



(그림 4) 구동부 전체 모습

체스 말 바닥에 소형 페라이트 자석을 같은 극성이 바닥면에 위치하도록 부착하였다. 초기에는 네오디움 자석으로 시도해 보았으나 자력이 너무 강하여 나이트의 행마법(行馬法)처럼 말 사이를 이동하는 경우 말끼리 영향을 주었다.



(그림 5) 상판을 올려 말을 배치한 모습

상판은 우드락과 아크릴로 구성되어있으며 약7mm의 두께를 가지고 있다. 전자석은 출력을 모터드라이버 칩을 이용하여 적절한 힘을 발휘하도록 조절하였다.

3.2 소프트웨어 구현

3.2.1 좌표 표현

제어부에서 구동부로 건네주는 좌표형식은 정식 체스경기에서 표기하는 방식을 따른다. 가로칸에는 a~h의 알파벳을 차례대로 붙이고 세로 칸에는 1~8까지 숫자를 붙여 좌표를 표현한다. 이때 사용자가 이동하는 경우를 4개의 문자로 표현하게 된다. 앞의 두 문자는 말의 처음위치, 뒤의 두 문자는 말의 나중 위치를 나타낸다. 예를 들어 'd2d4'는 d2에 위치한 말을 d4로 이동시켰다는 뜻이다. 구동부에서는 받아들이는 좌표를 모두 숫자로 바꾸어 순서대로 Xf,Yf,Xl,Yl로 저장을 한다. 나이트의 특수한 행마법을 구현하기 위해 칸과 칸 사이에 포인트를 지정하여 8*8칸이 아닌 16*16칸의 포인트를 가지도록 설정하였다. 따라서 수신된 좌표의 a~h는 2,4,6,…,16으로 대응되고 1~8도 2,4,6,…,16으로 대응되어 입력된 좌표가 X_{first},Y_{first},X_{last},Y_{last}에 최종적으로 저장이 된다.

3.2.2 제어부

제어부 프로그램은 Micro-Max 체스엔진[9]을 기반으로 하는 Chessuino[10] 프로그램을 수정하여 제작되었다. 기존의 Chessuino에서 블루투스 송수신 기능, 좌표용 스위치 변경, 그리고 3.2.4절에서 기술하는 구동부의 피드백을 처리하는 코드가 추가, 수정 되었다.

3.2.3 좌표 값에 따른 구동부 제어 알고리즘

Algorithm : 'Move X motor to ( )' function

Input : X_{goal}, $X_{current}$ Output : Void

```
1   if   X_current < X_goal
2   |    while   X_current != X_goal
3   |    |       Move X motor clockwise
4   |    end
5   else if   X_current < X_goal
6   |    while   X_current != X_goal
7   |    |       Move X motor counter clockwise
8   |    end
9   else
10  end
```

(그림 6) 목표좌표로 이동시키는 함수

그림. 6의 알고리즘은 Y축 모터에도 동일하게 적용된다.

$X_{current}$는 X축 광센서 모듈로부터 받은 모터의 현재 위치를 나타낸다. 실제 코드에서는 현재위치 값을 받는 함수가 별도로 존재하며 이를 모터가 이동하는 동안 지속적으로 호출하여 동작하지만 알고리즘 상에서는 하나의 입력변수로 표현하였다.

제어부로부터 좌표를 받으면 우선 X_{first}, Y_{first} 위치로 모터를 이동시키고 전자석을 켠다(Solenoie(on)). 그 이후 좌표 값에 따라 직선이동인지, 대각선 이동인지, 나이트의 이동인지를 파악한 뒤 그에 맞는 알고리즘에 따라 모터와 전자석이 이동하여 말을 움직인다. 이동이 완료한 뒤에 전자석을 끄고(Solenoid(off)) 제어부로 완료확인 문자를 전송한다.

Algorithm : 직선 이동

Input : X_{first}, Y_{first}, X_{last}, Y_{last} Output : Void

```
1   if   Y_first = Y_last   and   X_first   != X_last
2   |    Move X motor to (X_last)
3   end
4   if   X_first = X_last   and   Y_first   != Y_last
5   |    Move Y motor to (Y_last)
6   end
```

(그림 7) 말을 직선으로 움직이는 경우

대각선 이동방법은 좌상, 우상, 좌하, 우하 이렇게 4가지이다. X축과 Y축이 이동해야 할 거리는 같지만 각 축의 두 모터가 정확히 동일한 속도로 움직이지 않을 수 있기 때문에 Move X motor to() 함수를 사용하지 않고 for문을 통해 두 축 모두 한 포인트씩 이동을 완료했을 경우 다음 포인트로 이동하는 방식을 사용하였다. 그림. 8에 표기한 알고리즘은 우상(右上)방향으로 이동할 때의 경우이다.

Algorithm : 대각선 이동

Input : X_{first}, Y_{first}, X_{last}, Y_{last} , $X_{current}$, $Y_{current}$ **Output : Void**

```
1   if  abs(X_last - X_first) = abs(Y_last - Y_first)
2   |   if  (X_last - X_first) > 0 and (Y_last - Y_first) > 0
3   |   |   Move X motor clockwise
4   |   |   Move Y motor clockwise
5   |   |   for ( j=1; j≤(X_last-X_final); )
6   |   |   |   if  X_current - X_first = j
7   |   |   |   |  Turn off motor X
8   |   |   |   end
9   |   |   |   if  Y_current - Y_first = j
10  |   |   |   |  Turn off motor Y
11  |   |   |   end
12  |   |   |   if X_current - X_first = j and Y_current - Y_first = j
13  |   |   |   |  Turn off motor X and Y
14  |   |   |   end
15  |   |   |   j←j+1
16  |   |   end
17  |   end
18  end
```

(그림 8) 말을 대각선으로 움직이는 경우

Algorithm : 나이트 이동

Input : X_{first}, Y_{first}, X_{last}, Y_{last} **Output : Void**

```
1   if  abs(X_last - X_first) = 2 and abs(Y_last - Y_first) = 4
2   |   if  X_last - X_first  > 0
3   |   |  Move X motor to (X_first+1)
4   |   else
5   |   |  Move X motor to (X_first-1)
6   |   end
7   |   Move Y motor to (Y_last)
8   |   Move X motor to (X_last)
9   end
```

(그림 9) 나이트를 움직이는 경우

나이트의 경우 독특한 위치로 움직이게 된다. 구동방식의 한계로 인해 말을 뛰어 넘을 수 없으므로 말과 말 사이를 지나가는 방식을 사용하였다. 3.2.1절에서 좌표 포인트를 16*16으로 한 이유가 이것이다. 그림. 9에 기술된 알고리즘은 나이트가 X축으로는 1칸, Y축으로 2칸을 이동하는 경우를 나타낸다.

3.2.4 제어부와 구동부 간의 통신

Algorithm : 제어부와 구동부 간의 통신

```
1  '제어부' send 'S' to '구동부'
2  while Ture
3  |  Receive 'user coordinate'
4  |  Send user coordinate to '구동부'
5  |  while( !( receive 'E' from '구동부'))
6  |  end
7  |  if   Game over=True
8  |  |  print 'game over'
9  |  end
10 |  Calculate next coordinate
11 |  Send COM coordinate to '구동부'
12 |  while( !( receive 'E' from '구동부'))
13 |  end
14 |  if   Game over=True
15 |  |  print 'game over'
16 |  end
17 end
```

(그림 10) 제어부와 구동부 간의 통신

그림. 10에 제어부와 구동부 사이에 전송되는 데이터를 표현하였다. 처음 두 기기가 연결되면 확인을 위해 문자'S'를 보내며 말을 이동하고 난 뒤에는 문자'E'를 보내어 확인한다.