프로젝트 최종 보고서

[A 분반 4 조] 1972026 안하린, 1972027 여현영, 1972055 황예나

[B 분반 8 조] 1870076 임우정, 1876385 조송희, 1972031 윤희준

지도교수 : 박형곤 교수님

이화여자대학교 엘텍공과대학 전자전기공학과 2 팀

Abstract— **아두이노를 활용해 8 가지 서보모터 동작 유형을 자이로 센서로 인식하여 XBee 통신을 통해 송수신하고, 동작 패턴 인식 및 시각화하는 시스템을 구현하였다.**

정해진 총 8 가지 동작에 따라 2 개의 서보 모터가 동작하며 모터의 동작은 자이로 센서를 사용해 인식한다. 송신부에서 사용자가 연속적으로 실행시키는 동작을 구분하여 인식하기 위해 터치 센서를 이용하였다. 자이로 센서가 인식한 값은 무선 통신 모듈인 XBee 를 통해 전송되고, 수신부에서는 받은 자이로 센서 값을 통해 모터 동작을 구분하여 동작 패턴을 LCD 모니터로 시각화하였다.

I. INTRODUCTION

오늘날 품질 향상 및 다품종 소량생산에 대한 소비자 요구가 크게 증가하였다. 그러나 높은 인건비와 노동자의 삶의 질 향상 등을 고려하여 볼 때 급변하는 소비자의 요구를 기존의 공장에서와 같이 노동자의 힘으로만 충족시키기에는 어려움이 존재한다. 최근에는 이 문제점을 효과적으로 해결하기 위해 스마트 팩토리 구축에 주목하고 있다. 스마트팩토리는 제품의 기획, 설계, 생산, 유통 · 판매 등 전 과정이 사물인터넷(IoT, Internet of Things), 사이버 - 물리 시스템(CPS, Cyber - Physical System), 임베디드 운영 체제(ex. IoS) 등의 ICT 와 융합되어 자동화, 최적화된 프로세스로 고객 맞춤형 제품을 생산하는 팩토리를 말한다. 이러한 스마트 팩토리를 구현하기 위해서는 스마트 팩토리 내의 장비로부터 획득한 정보의 무선 통신 및 상황 실시간 감지 기술을 바탕으로 한 공정 모니터링 구현이 필수적이다. 본 프로젝트에서는 이를 위한 기반 기술으로서 모터의 동작에 따른 데이터를 무선으로 송수신하여 데이터를 받은 수신 측에서 동작을 판별하고자 한다.

II. BACKGROUND

A. Arduino

본 프로젝트에서는 Arduino Uno 의 32 비트 확장판인 Arduino Zero 를 사용한다.

B. Servo Motor

본 프로젝트에서는 Up, Down, Left, Right 를 비롯한 8 가지 동작 패턴을 구현해야 한다. 이를 위해 각각 0~180 도의 동작 범위를 가지는 서보모터 2 개를 함께 사용하여 최종적으로 3 차원 동작을 구현한다.

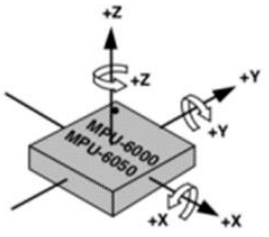
C. Touch Sensor

Touch Sensor 는 전자 이동을 이용하여 터치 유무를 확인하는 센서이다. 이 실험에서 활용한 TTP233 센서는 정전용량(capacitive) 방식으로 작동하는 터치 감지 IC 로 사용자의 몸에 흐르는 약한 전기를 이용하여 사용자가 터치 센서에 접촉하는 순간 정전용량이 변화하는 것을 감지하여 터치 유무를 감지하는 원리로 작동한다. 본 프로젝트에서는 Touch Sensor 의 터치 유무를 활용하여 송신할 데이터를 구분하고 개별적으로 송신하도록 함으로써 송신 측에서 진행한 연속된 서보모터의 움직임을 수신 측에서 구분하기 위해 사용하였다.

D. Gyro Sensor

자이로 센서는 물체의 가속도와 각속도를 측정하는 센서다. 본 프로젝트에서는 서보모터에 부착하여 인식한 가속도와 각속도를 기준으로 동작 패턴을 구분하는 데에 사용한다.

본 프로젝트에서 활용한 MPU6050 은 x, y, z 축 Accelerometer 와 x, y, z 축 Gyroscope 로 이루어져 있어 AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ 값을 측정하고 출력한다. Accelerometer 값(AcX, AcY, AcZ)은 각 축을 기준으로 기울어지는 가속도를 측정한 값이다. Gyroscope 값(GyX, GyY, GyZ)은 각 축 기준으로의 회전 속도를 측정한 값이다. 반시계 방향으로 돌면 양수 값이 나오고, 빨리 회전할수록 더 큰 값이 나온다.



[그림 1] MPU6050 축 방향

E. *XBee*

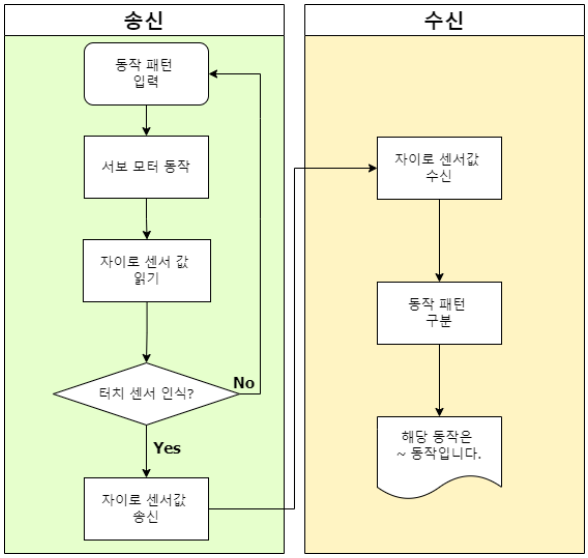
XBee 는 무선으로 시리얼 통신을 가능하게 하는 무선 데이터 통신용 모듈이다. 본 프로젝트에서는 총 2 개의 XBee 모듈을 활용하였다. XBee 모듈 1 개는 송신부에서 측정한 자이로 센서 값을 보내기 위해 사용하였고 다른 1 개는 수신부에서 송신부로부터 온 데이터를 받기 위해 사용하였다.

F. *LCD*

LCD 는 Adafruit 2.4 모델을 사용한다. 본 프로젝트에서는 해당 LCD 를 통해, XBee 로부터 수신한 동작 패턴 정보를 시각화하는 용도로 사용한다.

III. PROPOSED DESIGN

본 프로젝트는 [그림 2]와 같은 알고리즘으로 서보모터의 동작에서 감지되는 자이로 센서 값을 송수신하여 수행한 동작 패턴을 시각화하는 것을 목표로 한다.

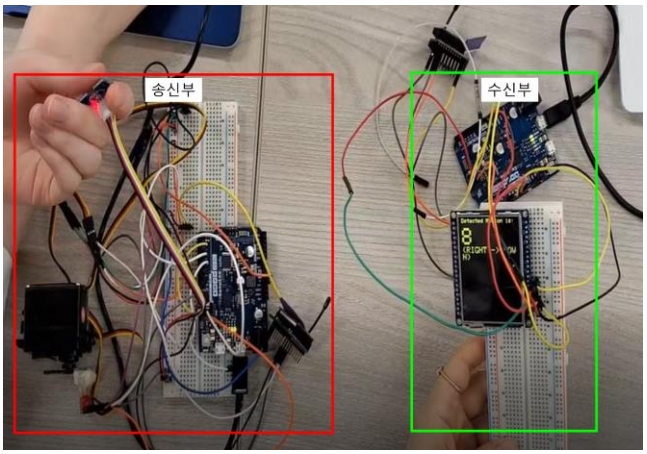


[그림 2] 프로젝트 전체 수행 알고리즘

A. *필수 구현 사항*

1. 시리얼 모니터를 이용해 원하는 동작을 입력한다. 동작이 입력되면 서보모터가 동작하며, 자이로 센서 값을 읽어온다.
2. 만약 터치 센서가 감지되면 읽어온 자이로 센서 값을 XBee 를 통해 송신한다. 그렇지 않으면 데이터를 송신하지 않고 다음 동작 입력을 기다린다.
3. 수신 측 아두이노에선 XBee 를 통해 자이로 센서 값을 받는다. 받은 데이터를 이용해 어떤 동작인지 인식하여 LCD 에 띄운다.

이를 구현한 최종 결과물 사진은 다음과 같다.

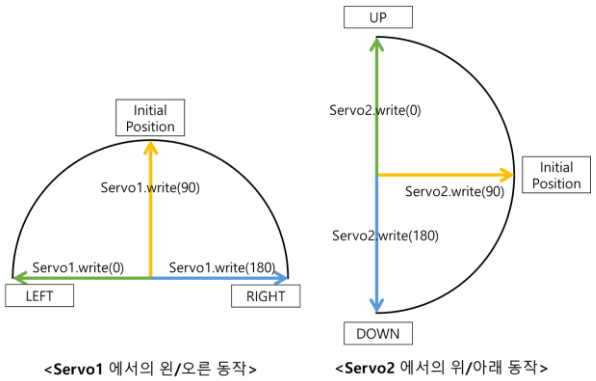


[그림 3] 최종 결과물

B. *동작 과정 설명*

1) *서보모터 제어*

송신부에서는 입력된 동작 패턴에 따라 서보모터를 동작시킨 뒤 동작이 끝났을 때를 터치 센서로 구분하여 측정된 데이터를 수신부에 전송한다.



[그림 4] 서보모터 입력 각도에 따른 위치

동작 패턴을 입력할 때는 아두이노의 시리얼 모니터에 동작과 매핑된 문자(a-h)를 입력한다. 아두이노 시리얼 모니터에 아래와 같은 동작 패턴에 해당하는 알파벳이 입력되면 해당 동작 패턴이 구현된다. 서보모터는 두 개를 연결한 구조로 설계하여 오른쪽 / 왼쪽 / 위쪽 / 아래쪽으로 움직일 수 있게 하였다. [그림 4]에서처럼 첫 번째 서보모터인 Servo1 은 오른/왼 이동을 담당하는 서보모터로, 0 을 입력할 경우 왼쪽, 180 을 입력할 경우 오른쪽을 가리키게 된다. 두 번째 서보모터 Servo2 는 위/아래 동작을 담당하여 0 을 입력하는 경우 위쪽, 180 을 입력하는 경우 아래쪽을 가리킨다. 따라서, 1~8 동작을 수행하기 위해선 아래 [표 1]과 같이 Servo1, Servo2 에 입력을 넣어줘야 한다. 모든 동작 후에는 위치 초기화를 위해 Servo1, Servo2 에 90 을 넣어주었다.

동작구분	동작 1	동작 2	Servo1 (오른/왼 이동담당)	Servo2 (위/아래 이동담당)
1 (a)	UP	-	90	0
2 (b)	DOWN	-	90	180
3 (c)	LEFT	-	0	90
4 (d)	RIGHT	-	180	90
5 (e)	UP	LEFT	90→0	0→0
6 (f)	UP	RIGHT	90→180	0→0
7 (g)	LEFT	DOWN	0→0	90→180
8 (h)	RIGHT	DOWN	180→180	90→180

[표 1] 동작에 따른 서보모터 입력값

2) *자이로 센서로 동작 감지하기*

입력 문자에 따라 서보모터를 제어하여 동작이 수행되면, 모터의 날개 한쪽에 부착된 자이로 센서를 통해 모터의 x 축 방향 가속도, y 축 방향 자이로 값, z 축 방향 자이로 값을 측정한다. 모터의 회전 방향에 근거하여 서로 다른 8 가지 동작을 모두 구분할 수 있도록 센서 값을 선정한 것이다. 연속된 동작이 완료되었는지를 구분하기 위하여, 터치 센서가 터치되어 있어야 센서 값이 수신부로 송신되도록 구현하였다. 센서 값 측정이 완료되었을 때 터치 센서에 입력이 감지되어 있다면, 측정된 센서 값을 문자열로 변환하여 XBee 를 통해 송신한다.

자이로 센서는 [그림 5]와 같이 상단 모터의 날개 한쪽에 부착되어 있다.

[그림 5] 자이로 센서 부착 위치

3) *XBee 를 이용해 송수신하기*

XCTU 프로그램을 실행시킨 후 XBee Dongle 을 사용해 XBee 의 초기 설정을 해준다. ID 는 개인 영역 네트워크 PAN ID 를 뜻하는 것으로, 서로 통신하는 XBee 는 같은 PAN ID 를 가져야 한다. MY 는 프로그래밍하는 사용자의 XBee 무선 모듈의 주소를 뜻하는 것으로, PAN 내의 모든 XBee 들은 고유한 주소 값을 가진다. DL 은 목적지 주소를 뜻하는 것으로, XBee 가 통신하려는 대상 모듈의 주소이다. 따라서 본 프로젝트에서는 [그림 6]과 같이 송신부와 수신부의 PAN ID 를 A424 로 통일시켜주었다. 그다음 MY 에 각각의 주소 즉 5678, 1234 를 설정한 뒤 DL 에 서로의 주소 즉 1234, 5678 로 설정해 주었다.

송신부 XBee 설정 수신부 XBee 설정

[그림 6] XCTU 를 이용한 XBee 설정

4) *수신된 자이로 센서 값 바탕으로 동작 구분하기*

수신부에선 XBee 를 이용하여 문자열로 자이로 센서 값을 수신하고 해당 값을 이용해 1~8 중 어느 동작 패턴인지 분류하여 나타내는 과정을 수행한다. 동작을 구분하는 기준은 다음과 같다.

- angleAcX1: 첫 번째 동작의 AcX 값
- angleGyY1: 첫 번째 동작의 GyY 값
- angleGyZ1: 첫 번째 동작의 GyZ 값
- angleAcX2: 두 번째 동작의 AcX 값
- angleGyY2: 두 번째 동작의 GyY 값
- angleGyZ2: 두 번째 동작의 GyZ 값

자이로 센서 측정값		동작 분류
첫 번째 자이로 센서 측정값	두 번째 자이로 센서 측정값	
0 < angleAcX1< 20	angleAcX2=0*	1(a) : UP
-20 < angleAcX1 < 0	angleAcX2=0*	2(b) : DOWN
-90 ≦ angleAcX1≦ -70, angleGyY1 < 0	angleAcX2=0*	3(c) : LEFT
-90 ≦ angleAcX1 ≦ -70, angleGyY1 > 0	angleAcX2=0*	4(d): RIGHT
0 < angleAcX1<20	angleGyZ2 < 0	5(e) : UP → LEFT
0 < angleAcX1<20	angleGyZ2 > 0	6(f) : UP → RIGHT
-90 ≦ angleAcX1 ≦ -65, angleGyY1 < 0	-20 < angleAcX2 < 0	7(g) : LEFT → DOWN
-90 angleAcX1 -65, angleGyY1 > 0	-20 < angleAcX2 < 0	8(h) : RIGHT → DOWN

[표 2] 자이로 센서 측정값에 따른 동작 구분표

* 두 번째 측정값이 존재하지 않음

a) 동작1(a): UP

(0 < angleAcX1 && angleAcX1 < 20 && angleAcX2 == 0)

UP 동작을 수행하면 위쪽 서보 모터가 위쪽으로 올라간다. 이는 x 축 기준으로 반시계 방향의 회전이기에 angleAcX1 값이 양수 값으로 나온다. 실험 결과 UP 동작에서는 angleAcX1 이 0 에서 20 까지의 범위로 나와 기준을 0 < angleAcX1 < 20 로 설정해 주었다.

b) 동작2(b): DOWN

(-20 < angleAcX1 && angleAcX1 < 0 && angleAcX2 == 0)

DOWN 동작을 수행하면 위쪽 서보 모터가 아래쪽으로 내려간다. 이는 x 축 기준으로 시계 방향의 회전이기에, UP 동작과는 반대로 angleAcX1 값이 음수 값으로 나온다. 실험 결과 DOWN 동작에서는 angleAcX1 이 -20 에서 0 까지의 범위로 나와 기준을 -20 < angleAcX1 < 0 로 설정해 주었다.

c) 동작3(c): LEFT

(-90 <= angleAcX1 && angleAcX1 <= -70 && angleGyY1 < 0 && angleAcX2 == 0)

LEFT 동작을 수행하면 아래쪽 서보 모터가 왼쪽으로 돌아간다. 이는 y 축 기준으로 시계 방향의 회전이기에 angleGyY1 값이 음수 값으로 나온다. 실험 결과 LEFT 동작에서는 angleAcX1 이 -90 에서 -70 까지의 범위로 나와 1, 2 번 동작과 구분하고자 기준을 -90 < =angleAcX1 <= -70 로 설정해주었고, angleGyY1<0 의 기준을 주었다.

d) 동작4(d): RIGHT

(-90 <= angleAcX1 && angleAcX1 <= -70 && angleGyY1 > 0 && angleAcX2 == 0)

RIGHT 동작을 수행하면 아래쪽 서보 모터가 오른쪽으로 돌아간다. 이는 y 축 기준으로 반시계 방향의 회전이기에 angleGyY1 값이 양수 값으로 나온다. 실험 결과 RIGHT 동작에서는 angleAcX1 이 -90 에서 -70 까지의 범위로 나와 1, 2 번 동작과 구분하고자 기준을 -90 < =angleAcX1 <= -70 로 설정해주었고, angleGyY1>0 의 기준을 주었다.

1,2,3,4 동작 모두 두 번째 동작이 없기 때문에 angleAcX2==0 이라는 조건을 주었다.

e) 동작5(e): UP->LEFT

(0 < angleAcX1 && angleAcX1 < 20 && angleGyZ2 < 0)

먼저 UP 동작을 수행하면 위쪽 서보 모터가 위쪽으로 올라간다. 이는 x 축 기준으로 반시계 방향의 회전이기에 angleAcX1 값이 양수 값(0 과 20 사이의 값)으로 나온다.

그 상태에서 LEFT 동작을 수행하면 아래쪽 서보 모터가 왼쪽으로 돌아간다. UP 동작으로 인해 축이 바뀌어서 이는 z 축 기준으로 시계 방향의 회전이 되어 angleGyZ2 값이 음수로 나온다.

f) 동작6(f): UP->RIGHT

(0 < angleAcX1 && angleAcX1 < 20 && angleGyZ2 > 0)

먼저 UP 동작을 수행하면 위쪽 서보 모터가 위쪽으로 올라간다. 이는 x 축 기준으로 반시계 방향의 회전이기에 angleAcX1 값이 양수 값(0 과 20 사이의 값)으로 나온다.

그 상태에서 RIGHT 동작을 수행하면 아래쪽 서보 모터가 오른쪽으로 돌아간다. UP 동작으로 인해 축이 바뀌어서 이는 z 축 기준으로 반시계 방향의 회전이 되어 angleGyZ2 값이 양수로 나온다.

g) 동작7(g): LEFT->DOWN

(-90 <= angleAcX1 && angleAcX1 <= -65 && angleGyY1 < 0 && -20 < angleAcX2 && angleAcX2 < 0)

LEFT 동작을 수행하면 아래쪽 서보 모터가 왼쪽으로 돌아간다. 이는 y 축 기준으로 시계 방향의 회전이기 때문에 angleGyY1 값이 음수 값으로 나온다. 실험 결과를 바탕으로 LEFT 동작에서는 기준을 -90 < =angleAcX1 <= -65 로 설정해주었고, AngleGyY1<0 의 기준을 주었다.

이후 DOWN 동작을 수행하면 위쪽 서보 모터가 아래쪽으로 내려간다. 이는 x 축 기준으로 시계 방향의 회전이므로 angleAcX2 값이 음수 값으로 나온다. 실험 결과를 바탕으로 DOWN 동작에서는 기준을 -20 < angleAcX2 < 0 로 설정해 주었다.

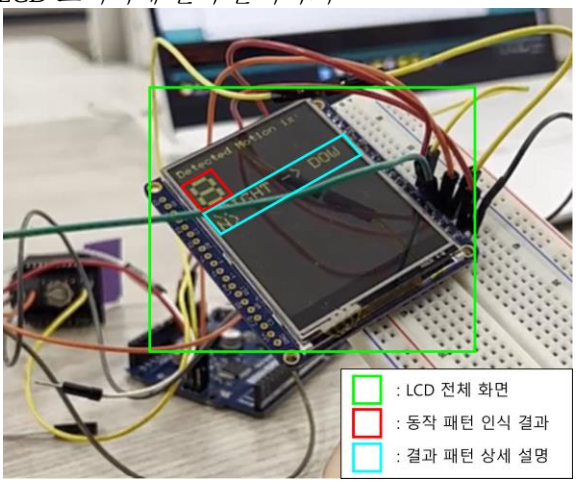
h) 동작8(h): RIGHT->DOWN

(-90 <= angleAcX1 && angleAcX1 <= -65 && angleGyY1 > 0 && -20 < angleAcX2 && angleAcX2 < 0)

RIGHT 동작을 수행하면 아래쪽 서보 모터가 오른쪽으로 돌아간다. 이는 y 축 기준으로 반시계 방향의 회전이기 때문에 angleGyY1 값이 양수 값으로 나온다. 실험 결과를 바탕으로 RIGHT 동작에서는 기준을 -90 < =angleAcX1 <= -65 로 설정해주었고, angleGyY1>0 의 기준을 주었다.

이후 DOWN 동작을 수행하면 위쪽 서보 모터가 아래쪽으로 내려간다. 이는 x 축 기준으로 시계 방향의 회전이기 때문에 angleAcX2 값이 음수 값으로 나온다. 실험 결과를 바탕으로 DOWN 동작에서는 기준을 -20 < angleAcX2 < 0 로 설정해 주었다.

5) LCD 모니터에 결과 출력하기



[그림 7] LCD 출력

[그림 7]과 같이 LCD 로 동작 패턴 인식 결과를 출력한다. LCD 화면에 동작 패턴 인식 결과를 나타내는 번호(1~8 의 정수)와 함께, 해당 동작 패턴의 상세 동작을 설명하는 문구가 출력된다.

IV. CODE DESPCRITION

A. 서보 모터 움직이기 및 자이로 센서 인식하기

```
void loop()
{
  if(Serial.available()>0)
  {
    in_data = Serial.read();

    String sensor = "";


    if(in_data == 'a'){

      servo2.write(0);
      delay(1000);
      Serial.print("input = ");
      Serial.println(in_data);
      getData();
      getAngle();
      S_angleAcX=String(angleAcX);
      S_GyY=String(GyY);
      S_GyZ=String(GyZ);

      sensor=S_angleAcX+"/"+S_GyY+"/"+S_GyZ+"/0/0/0";

      //초기화
      servo1.write(90);
      delay(1000);
      servo2.write(90);
      delay(1000);
```

[그림 8] 서보모터 제어 및 자이로 센서 인식 code

먼저, 시리얼 창에 입력된 값을 in_data 라는 변수로 받아 해당 문자에 따라 동작이 이루어진다. 시리얼 창에 “a”가 입력되면 Servo.write() 함수를 이용하여 서보 모터가 해당 동작을 수행한다.

서보 모터가 동작을 수행하면 getData() 함수를 통해 자이로 센서가 그 동작으로 인한 AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ 값을 측정한다. 동작 수행과 getData() 코드 사이에 delay 를 준 것은 모터의 동작 시작 시점과 센서의 측정 시점 간 시간 간격을 주기 위해서이다. 한 종류의 동작 당 자이로 센서 측정값을 한번 얻게 되는데 2 개의 동작이 연속적으로 이어져 있는 5, 6, 7, 8 의 경우 첫 번째 동작을 할 때의 자이로 센서 측정값, 두 번째 동작을 할 때의 자이로 센서 측정값을 각각 입력받는다. 모든 동작이 완료되면 Servo1, Servo2 를 모두 90 으로 초기화시킨다.

B. 터치 센서와 XBee 통신

```
int sensorValue=digitalRead(TouchPin);
if(sensorValue==1)
{
  sensor.toCharArray(buf,sensor.length()+1);
  Serial2.write(buf); //송신
  Serial.println(buf);
```

[그림 9] 터치 센서와 XBee 송신 code

입력 값에 따라 모터가 동작하고 각 동작에 대한 자이로 센서 값이 측정되어 sensor 라는 string 에 각 동작별 센서 값이 저장되고 나면, 사용자와의 터치 센서 접촉 여부를 확인한다. XCTU 를 이용해 설정한 XBee 를 아두이노에 연결하고, 만약 사용자가 터치 센서를 접촉하고 있으면 sensorValue 가 1(High)이 되어 수신부로 sensor string 을 char array 로 변환한 buf 를 Serial2.write(buf)을 이용해 자이로 센서 값을 송신한다.

사용자가 수신부로 해당 동작을 전송하고자 하지 않는다면 터치 센서와 접촉하지 않으면 된다. 이 경우에는 loop( ) 함수의 앞부분으로 돌아가 다시 모터를 동작하기 위한 입력을 받도록 대기하는 상태가 된다. 즉, 터치 센서를 통해 송신부에서 수행한 동작 중 수신부로 전송하고자 하는 동작을 선별하고 송신부에서 진행한 각각의 동작에 대한 개별적인 정보를 보내게 함으로써 연속된 모터의 동작을 구분할 수 있게 된다.

C. *Xbee 수신*

```
//receive data 받아와서 string 생성
String makeString()
{
  String readString;
  while (Serial2.available())
  {
    delay(2);
    char c = Serial2.read();
    readString += c;
  }
  if (readString.length() > 0)
  {
    Serial.println(readString);
  }
  return readString;
}
```

[그림 10] XBee 수신 code

수신부에서는 Serial2.read() 함수를 이용하여 송신부에서 보낸 값을 읽어온다. 이때, char type 은 한 문자만 읽을 수 있으므로 while 문을 사용하여 얻어온 문자들을 string type 으로 합쳐 송신부에서 보낸 모든 값을 받는다.

D. *자이로 센서 값으로 동작 구분하기*

```
//들어온 자이로센서값 따라 모션 분류
char motionClassification(double angleAcX1, double angleGyY1, double angleGyZ1,
                          double angleAcX2, double angleGyY2, double angleGyZ2)
{
  char result;

  if ( 0 < angleAcX1 && angleAcX1 < 20 && angleAcX2 == 0)
  {
    result = 'a';
  }

  else if (  -20 < angleAcX1 && angleAcX1 < 0 && angleAcX2 == 0)
  {
    result = 'b';
  }

  else if (  -90 <= angleAcX1 && angleAcX1 <= -70 && angleGyY1 < 0 && angleAcX2 == 0)
  {
    result = 'c';
  }

  else if (   -90 <= angleAcX1 && angleAcX1 <= -70 && angleGyY1 > 0 && angleAcX2 == 0)
  {
    result = 'd';
  }
```

[그림 11] 동작 a~d 구분 code

수신 측에서 자이로 센서 값들과 미리 설정한 기준(III.B.4 참고)을 바탕으로 동작 구분을 수행한다.

V. DISSCUSSION

A. *Conclusion*

본 프로젝트에서는 정해진 8 개의 패턴에 따른 로봇의 움직임을 감지하여 실시간 무선 통신을 통해 동작 패턴을 모니터링할 수 있는 시스템을 구현하였다. 자이로 센서 값을 통해 스마트팩토리에서 로봇의 동작을 자동적으로 인식함으로써, 인간 노동자의 개입을 최소화하고 설비 관리 및 제어를 효율화하는 데에 기여할 수 있다.

B. *한계점*

첫째, 서보 모터의 동작 패턴을 분류하는 알고리즘이 실제 상황에서는 제한적 성능을 보일 가능성이 있다. 본 프로젝트에서 사용된 동작 패턴 인식 알고리즘은 제한적인 실험 상황에서의 각 동작의 자이로 센서 값을 기준으로 결정하였다. 실험 상황에서 제안한 알고리즘은 정확도를 유지하지만, 실제 상황에서는 프로젝트 조건에 제시된 8 가지 동작 이외의 다양한 돌발적인 동작 패턴이 나타날 수 있다. 또한 실제 상황에서는 장비의 운행 중 떨림, 센서의 민감도 등으로 인해 측정과정에서 잡음이 발생할 가능성이 있다. 이러한 실제 상황에서도 오작동이나 위험 상황을 인지하는 것은 스마트팩토리 운용의 안정화를 위해 필수적이다. 따라서 실제 로봇의 동작 패턴을 인식하는 데에 사용된다면 자이로 센서를 통해 인식한 데이터를 대량으로 축적, 학습하여 딥러닝 기반 분류 모델을 설계함으로써 분류를 위한 기준 및 알고리즘을 고도화하여 정확도를 높여야 한다.

둘째, 터치 센서를 통해 모터의 연속된 동작을 구분하는 방법은 결국 사용자의 판단이 들어가야하므로 실제 스마트팩토리 환경에서 바로 적용하기 어렵다. 한 동작이 완결되면 로봇 자체에서 단일 동작 종결을 알리는 신호를 보내는 등 측정하는 기기 자체적으로 동작의 시작과 끝을 구분할 수 있도록 하는 개선이 필요하다.

셋째, 본 프로젝트에서 사용한 서보 모터는 180 도까지만 움직일 수 있어 UP, DOWN, LEFT, RIGHT 동작을 하나의 서보 모터에서 구현할 수 없다. 360 도 회전할 수 있는 서보 모터를 사용하면 모든 동작을 하나의 서보 모터에서 구현할 수 있어 더 다양한 동작을 수행할 수 있다.

VI. REFERENCES

[1] http://kocw.xcache.kinxcdn.com/KOCW/document/2018/bufs/limintaek271/12.pdf

[2] 융합연구리뷰 Convergence Research Review 2020 December vol.6 no.12, 한국과학기술연구원 융합연구정책센터, 2020 년 12 월