Journal of The Korea Society of Computer and Information
Vol. 23 No. 8, pp. 17-23, August 2018

www.ksci.re.kr
https://doi.org/10.9708/jksci.2018.23.08.017

Implementation of Flight Simulator using 6DOF Motion Platform

Myeong-Chul Park*, Duk-Kyu Choi**

Abstract

In this paper, we implemented a flight posture simulator that intuitively understands aircraft flight posture and visualizes the principle of motion. The proposed system operates the 6 - axis motion platform according to the change of the navigation information and transmits the flight attitude to the simulator using the gyro sensor. A gyro sensor and an acceleration sensor are used together to analyze the attitude of the aircraft. The reason is that the gyro sensor has a cumulative error in the integration process. And the accelerometer sensor was compensated by using the complementary filter because noise was serious due to short term vibration. Using the compensated sensor information, the motion platform is operated by calculating the angle to be transmitted to the 6-axis motor. And visualization result is implemented using OpenGL. The results of this study can be used as teaching materials for students related to aviation in the future.

▸Keyword: Flight simulation, Motion platform, gyro sensor, 6DOF Motor

I. Introduction

비행 시뮬레이션은 가상현실 기술을 사용하는 산업의 한 분야로 자리 잡았다. 비행 시뮬레이션은 실제 항공기 없이 민간 또는 군사 조종사를 훈련시키는 데 사용할 수 있지만 정교한 컴퓨터 시스템인 비행 시뮬레이터를 교육하는 데 사용할 수도 있다. 가상현실 또는 가상 환경 시스템은 일반적으로 사용자가 컴퓨터와 대화식으로 인터페이스하고 3D 시각적 작업에 적용되는 시스템입니다. 이 시스템은 객체 또는 완전한 환경의 3D 모델을 지원하고 변환에 적합한 가상 환경을 제공하므로 사용자는 실시간으로 시스템과 상호 작용할 수 있습니다. 몰입감, 실시간 상호 작용, 3D 그래픽 및 포스 피드백(Force Feedback)과 같은 기능은 잘 알려진 시스템 개념이며 오래전부터 비행시뮬레이션에서 사용되어져 왔다. 하지만 대부분의 관련 시스템은 비정상적인 작동 조건에서 항공기를 다루는 기술 및 조종사 등의 전문가 집단을 위한 훈련 도구로 사용되었다 [1-4]. 본 연구에서는 항공 및 항적정보의 이해가 부족한 학생들을 위하여 아두이노 기반의 6축 모션 플랫폼을 통한 비행 자세 시뮬레이션을 구현한다. 먼저, 입력된 항적정보를 이용하여 6축 모션 플랫폼의 동작시킨다[5].

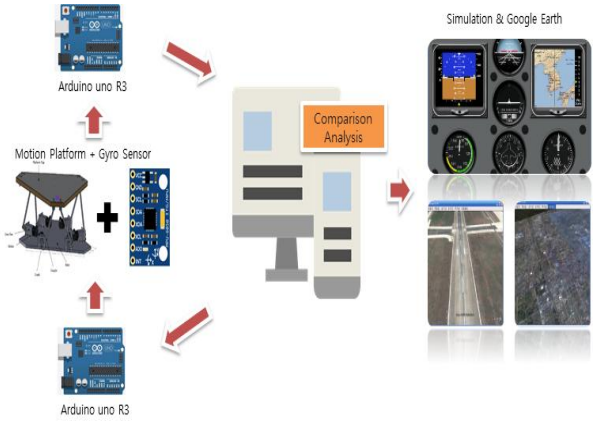

Fig. 1. Diagram of 6DOF Motion Platform & Simulation

모션 플랫폼은 Yaw, Pitch, Roll 값을 입력으로 하여 6축의 서보모터에게 전송될 각각의 각도를 계산한다. 모션 플랫폼이 동작하게 되면 부착된 자이로 센서[6-7]에 의해 변화 값이 2차 아두이노에 전송되고 시뮬레이션을 위한 미션 컴퓨터에 자

• First Author: Myeong-Chul Park, Corresponding Author: Duk-Kyu Choi
*Myeong-Chul Park (africa@ikw.ac.kr), Dept. of Avionics Engineering, KyungWoon University
**Duk-Kyu Choi (dkchoi@ikw.ac.kr), Dept. of Avionics Engineering, KyungWoon University
• Received: 2018. 07. 31, Revised: 2018. 08. 10, Accepted: 2018. 08. 13.
• This paper is supported by KyungWoon University.

이로 값을 전송하게 된다. 마지막으로 미션 컴퓨터는 기존의 항적정보와 자이로 값을 비교분석하여 최종적인 시각화를 위한 비행계기 시뮬레이션과 Google Earth를 동작시키게 된다. 시뮬레이션은 OpenGL[8]과 C++언어로 구현하였고 자이로 센서의 출력 값에 노이즈가 많아 실제 자세정보와 불일치하여 상보필터를 사용하여 보정하였다. 제안하는 시스템의 전체적인 구성은 Fig.1과 같다.

II. Background

항공기에 작용하는 힘은 크게 중력, 추력, 항력, 양력 등이 있는데 이 네 가지 힘의 작용성을 정확히 알아야 관련 시뮬레이션을 이해하고 제작할 수 있다. 중력(gravitiy)은 아시는 바와 같이 지구가 사물을 잡아당기는 힘을 의미한다. 중력은 그 자체를 측정할 수 없기 때문에 일반적으로 중력 가속도(gravitational acceleration)를 측정한다. 지구가 완벽한 구형이 아니기 때문에 위도가 낮을수록 중력가속도는 작아지는 등 지점에 따라 차이가 있지만 약 9.8m/s2 로 인식되고 있다.

Table 1. Secondary or Auxiliary Control Surfaces

Item	Function
Flaps	Extend wing camber to allow greater lift and slower flight
Trim tabs Balance tabs	Reduce the power required to move the main pilot plane
Anti-balance tabs	Increase the feel and efficiency of the main pilot
Servo tabs	Helps or provides the power to move the main flight control
Spoilers	Complement the aileron function.
Slats	Extend wing camber to allow greater lift and slower flight
Slots	Induce air through the top of the wing during high attack angles
Leading edge flap	Extend wing camber to allow greater lift and slower flight

추력(thrust)은 항공기의 프로펠러의 회전으로 공기를 밀어내거나 엔진의 분사에 의해 반대 방향으로 발생하는 추진력을 의미한다. 항력(drag)은 사물이 움직일 때 이 움직임에 저항하는 힘을 의미한다. 일반적으로 항력은 마찰력과 압력으로 구분되는데, 마찰력은 사물 표면에 수평적으로 작용하고 압력은 수직적으로 작용한다. 앞선 추력은 이러한 항력을 극복하고 비행하기 위한 힘을 의미한다. 양력(lift)은 사물이 수직방향으로 받는 힘으로 항공기 주위에 유체가 흐를 경우 표면에서 유체 흐름에 대하여 수직 방향으로 발생하는 힘인데 이 힘으로 항공기가 뜨게 된다. 고정익 항공기의 주요 비행 조종면은 에일러론(Ailerons), 엘리베이터(Elevators) 및 방향타(rudder)를 포함한다. 에일러론은 양쪽 날개의 후연에 부착되며, 움직일 때 항공기를 세로축 주위로 회전시킨다. 엘리베이터는 수평꼬리날개 끝에 부착된다. 움직일 때 수평 또는 측면 축에 대한 태도 인 항공기 피치가 변경된다. 러더는 수직꼬리날개 끝에 결합되어 있다. 방향타가 위치를 변경하면 항공기는 수직 축을 중심으로 회전한다. 그 외 보조 비행 조종면의 이름, 위치 및 기능은 아래 Table 1과 같다.

III. Design

1. Controlling the Motion Platform

모션 플랫폼은 6개의 가변 길이 커플러 연결된 2개의 단단한 프레임(Top Plate, Base Plate)으로 구성된다. Base Plate는 직교 축 x, y, z를 갖는 참조 프레임 워크로 간주된다. 플랫폼은 자체의 직교 좌표 x', y', z'를 가진다. 그리고 플랫폼은 Base Plate에 대해 6 자유도를 가진다. 플랫폼 좌표의 원점은 각 축마다 하나씩 Base Plate에 대한 3개의 선형 변위로 정의할 수 있다. 3개의 각도 변위는 Base Plate에 대한 플랫폼의 방향을 정의한다. 오일러 각의 한 집합은 z 축을 중심으로 각도 Ψ (Yaw)를 회전, y 축을 중심으로 각도 Θ (Pitch)를 회전, x 축을 중심으로 각도 Φ (Roll)를 회전 한다.

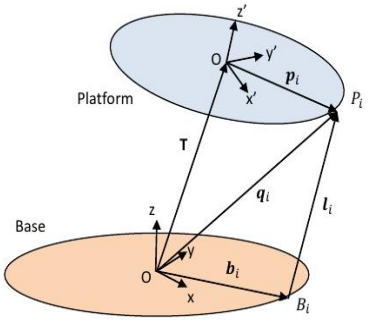

Fig. 2. Mathematical model of motion platform

Base reference framework에 대한 앵커 포인트 P_i 의 좌표 q_i는 다음 방정식에 의해 주어진다.

$$q_i = T + {}^{P}R_B \cdot p_i$$

여기서, T는 베이스 벡터를 기준으로 플랫폼 프레임의 원점의 위치적인 선형 변위를 제공하는 변환 벡터이고, p_i 는 플랫폼 프레임 워크와 관련하여 앵커 포인트 P_i 의 좌표를 정의하는 벡터이다. 유사하게 i 번째 커플러 길이는 다음과 같이 주어진다.

$$l_i = T + {}^{P}R_B \cdot p_i - b_i$$

여기서 b_i 는 벡터이다.

Forward Kinematics[9]를 고려할 때, 이 표현식은 플랫폼의 위치와 태도를 나타내는 6개의 미지수에서 18개의 동시 비

선형 방정식을 나타낸다. 이 방정식의 해를 찾는 데 많은 작업이 수행되어야 한다. 일반적인 경우에는 가능한 해결책이 여러 가지가 있지만 실제로는 이러한 솔루션 중 상당 부분이 실용적이지 않다. 커플러 길이가 선형 서보가 아닌 회전 서보를 통해 이루어지는 경우 서보의 회전 각도를 결정하려면 추가 계산이 필요합니다. 플랫폼에 대한 역기구학에 대해 유효한 "커플러"의 길이와 서보 암의 관련 각도를 계산할 수 있는 충분한 정보가 있다. 그러나 모션 플랫폼을 설계하고 구현하려면 이동 범위를 정의하기 위해 몇 가지 상수를 정의해야 한다.

1) 플랫폼의 "Home" 위치를 정의해야 한다. 정의에 따르면 플랫폼이 기본 프레임 워크보다 높은 위치에 있으며 다른 이동 또는 회전 동작이 없다.

$$q_i = T + {}^{P}R_B \cdot p_i$$

$$q_0 = \begin{bmatrix} 0 \\ 0 \\ h_0 \end{bmatrix} + \begin{bmatrix} 1\,0\,0 \\ 0\,1\,0 \\ 0\,0\,1 \end{bmatrix} \cdot \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}$$

$$= \begin{bmatrix} x_p \\ y_p \\ h_0 + z_p \end{bmatrix}$$

마찬가지로 서보 암과 로드가 서로 직각을 이루는 "Home" 위치를 정의한다.

$$l^2 = s^2 + a^2$$

$$= (x_p - x_b)^2 + (y_p - y_b)^2 + (h_0 - z_p - 0)^2$$

$$h_0 = \sqrt{s^2 + a^2 - (x_p - x_b)^2 - (y_p - y_b)^2} - z_p$$

플랫폼은 z 축을 중심으로 대칭으로 구성되기 때문에 이 방정식은 모든 커플러에 대해 동일한 결과를 제공한다.

2) Home 위치에서 서보 암의 각도도 계산할 수 있다.
다음 식을 이용하여

$$l_i = T + {}^{P}R_B \cdot p_i - b_i$$

플랫폼의 대칭 구조를 기억하면, "Home" 위치에 있는 커플러 길이는 다음과 같다.

$$l_0 = \begin{bmatrix} x_p \\ y_p \\ h_0 + z_p \end{bmatrix} - \begin{bmatrix} x_p \\ y_p \\ 0 \end{bmatrix}$$

$$l_0^2 = (x_p - x_b)^2 + (y_p - y_b)^2 + (h_0 + z_p)^2$$

"Home" 위치에서 서보 암의 각도는 다음 방정식에 의해 주어질 수 있다.

$$\alpha = \sin^{-1}\frac{L}{\sqrt{M^2 + N^2}} - \tan^{-1}\frac{N}{M}$$

대칭을 가지고 있으므로, 커플러 2개만 고려하면 된다. 여기서 β = 0° 이다.

$$\alpha_0 = \sin^{-1}\frac{L_0}{\sqrt{M_0^2 + N_0^2}} - \tan^{-1}\frac{N_0}{M_0}$$

$$where \;\; L_0 = l^2 - (s^2 - a^2)$$
$$= s^2 + a^2 - (s^2 - a^2) = 2a^2$$

$$M_0 = 2a[\cos\beta(x_p - x_b) + \sin\beta(y_p - y_b)]$$
$$= 2a(x_p - x_b)$$

$$N_0 = 2a(h_0 + z_p)$$

모션 플랫폼을 제어하기 위한 절차는 Fig.2와 같다. 먼저, 미션 컴퓨터와 자이로 센서로 부터 자세정보를 입력받아 보정작업을 위한 상보필터를 거친다. 그리고 회전 각도를 계산하고 적용할 각 모터의 의 축에 대한 위치 값을 계산한다. 그 결과 값을 통하여 모터에 전송될 회전 각도를 계산하게 된다. 그리고 각 서보모터에 대한 유지 시간을 계산하여 모터에 적용하게 된다.

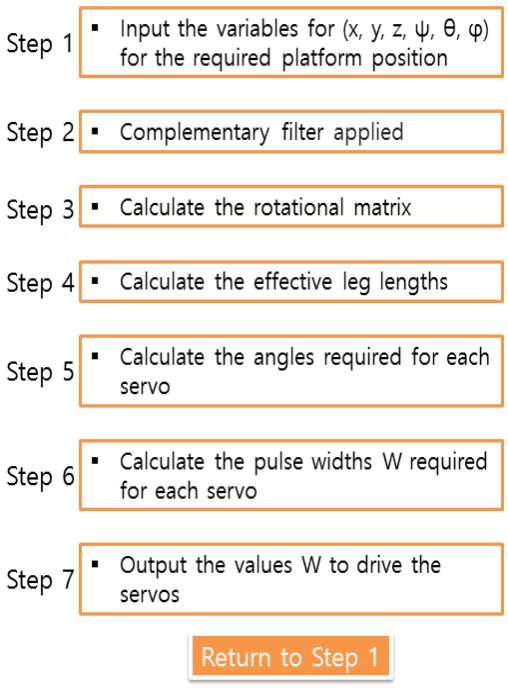

Fig. 3. Flowchart for Motion Platform Control

2. Complementary Filter

구현 시스템에서 가속도 값을 사용하는 이유는 각 방향의 기울어진 정도를 측정하는 것이다. 사용하는 MPU6050 센서는 Z축과 다른 축의 값을 atan2 연산을 통하여 측정이 가능하다. 하지만, 이 가속도 값만으로 각도를 측정할 경우 어떤 방향으로 움직여도 그 방향으로 가속도가 발생하기 때문에 각도를 측정하는 센서의 결과 값에 순간적인 노이즈가 발생하게 된다. 그리고 자이로 센서는 각속도를 측정하는데 이는 회전하는 속도를 의미한다. 이 값은 적분을 통해 얻어지는데 축적을 할수록 실제 각도와의 차이가 발생하게 된다[10]. 이러한 문제점을 해결하기 위하여 사용하는 것이 상보필터이고 그 식은 아래와 같다.

$$\angle = 0.98 * (\angle + gyro * dt) + 0.02 * (acc)$$

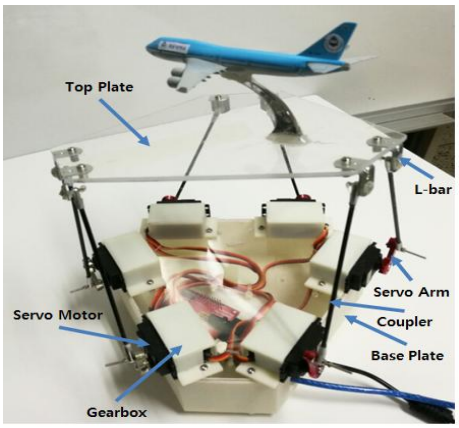

Fig. 4. 6DOF Motion Platform

여기서, $\angle$ 은 최종각도, gyro는 자이로센서값, dt는 적분시간, acc는 가속도를 의미한다. 상수 값으로 사용된 0.98과 0.02는 자이로 값과 가속도 값의 상대적인 중요성에 대한 변경 파라미터이다. 비행자세는 매순간 측정값에 따라 움직임으로 가속도보다 자이로 값이 더 중요하다고 볼 수 있다. 구현된 모션 플랫폼은 Fig. 3과 같다.

III. Implementation of Flight instruments

모션 플랫폼의 동작에 따른 자세정보를 다시 미션 컴퓨터에 전송하면 미션컴퓨터는 기본 항적정보와 비교하여 보정작업을 거쳐 최종적인 시뮬레이션 모듈로 자세정보 및 계기 정보를 전송하게 된다. 실제 구현된 시각화 결과는 Fig. 4와 같다. 비행자세 및 실제 환경을 가시화하기 위하여 OpenGL을 이용하여 계기 정보를 표시하였고 Google Earth를 이용하여 비행 상태를 시각화하였다. 또한 객관적인 위치정보를 보이기 위하여 오픈 맵을 이용하여 계기 정보에 DMM(Digital Moving Map)을 구현하였다.

Fig. 5. Simulation & Google Earth

Fig. 4 좌측의 MFD와 우측의 맵 정보로 이루어져 있고 중앙에 PFD(Primary Flight Display)에 대한 정보가 각각의 계기를 통해 보여 진다. 실제적인 이미지를 통하여 사실성을 증대하였다. 매핑 작업은 roll과 pitch값에 의해 중심좌표를 기준으로 회전좌표를 계산하여 해당 영역에 매핑하고 있다. OpenGL에서 제공하는 회전함수는 다른 이미지 영역에 영향을 주므로 별도의 회전 좌표를 구하여 이미지에서 해당 영역을 추출하여 매핑하는 방법을 사용하였다. Fig. 4 우측의 실시간 지도정보는 10단계 수준으로 구분되어 있으며 온라인 정보를 표시한다. 지도정보는 ArcGIS의 World Street Map[11]의 MapServer를 통해 가져오게 되는데 축척은 73,874,399 에서 144,285까지 지원한다. 상하 방향키를 누르면 축척의 수준이 자동으로 바뀌게 되고 맵의 우측에 해당 레벨이 표시되게 된다. 지도의 중앙에는 현재 비행하는 경위도의 좌표가 배치되고 최대 2,097,152(레벨 10)장의 분할된 지도 이미지를 해당 경위도에 맞게 화면에 배치하게 된다. 레벨 4에 해당하는 각각의 이미지는 경도와 위도가 11.25도에 해당하는 영역의 이미지를 가진다. 즉, 경도 영역으로 32장(360/11.25)의 이미지와 위도 영역으로 16장(180/11.25)의 이미지를 가진다. 가령, 경운대학교의 경도와 위도를 구글 지도에서 검색해 보면 경도는 128.4657047 이고 위도는 36.1694576 이다. 이를 다음 공식에 적용하면(zoom_ra = 11.25) x_stno와 y_stno는 각각 27과 4가 나온다. 이를 이용하여 해당 지도정보를 얻기 위한 URL을 취할 수 있다.

```
x_stno=abs((int)(((-180)-lon)/zoom_ra));
y_stno=abs((int)((lat-90)/zoom_ra));
```

*http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_StreetMap_World_2D/MapServer/tile/4/**4**/**27**.jpg*

Fig. 5의 (A)와 같이 9장의 이미지가 맵핑되어 있다고 가정하면 현재 비행하는 경위도(중심좌표)는 항상 [2.2]에 존재하게 한다. 그리고 해당 좌표를 맵상의 중앙에 위치하게 하고 나머지 이미지를 각각 부분적으로 해당 영역에 맵핑한다. 예를 들어 현재 비행하고 있는 경도와 위도가 [2.2]의 오른쪽 하단에 위치한다면 맵의 배치는 Fig. 5의 (B)와 같다.

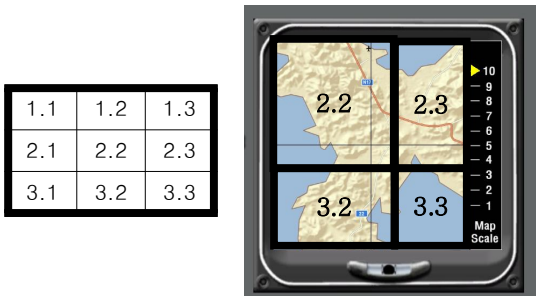

Fig. 6. Map image of DMM

Ⅳ. Interconnection in Google Earth

항적 정보를 비행계기와 Google Earth[12]에 전달하기 위해서 먼저, UDP(User Datagram Protocol) 패킷을 수신하기 위하여 패킷 포맷의 구조체를 정의한다. 하나의 패킷에는 인덱스를 포함하여 최대 8개의 독립된 데이터가 전달될 수 있다. 각각의 데이터는 4바이트 실수형으로 정의되어 있으며 단일 데이터만이 들어 있는 패킷 인덱스도 존재한다. UDP 데이터 취득을 위한 UDP 서버 프로그램 및 Google Earth 연동프로그램을 구현하여 UDP 데이터 취득을 위한 단위 기능을 시험하고 사실적인 데이터 확인한다. 구조체에 저장된 UDP 패킷정보는 조종석 모의 계기 모듈을 위하여 시각화 엔진에게 값을 전달한다. 시각화 엔진은 전달 받은 값을 계기를 시연하기 위한 목적으로 해당 단위 계기 모듈에게 해당 값을 전달한다. 값을 전달하기 전에 동작하는 단위 모듈에 특성에 따라 약간의 보정작업을 거쳐 전달한다. Fig. 6에서 좌측 이미지는 보정작업 이전의 Google Earth 화면이고 우측 이미지는 보정작업 이후의 보정된 Google Earth 화면이다. 보정작업 이전보다 사실적인 비행 Pitch 값이 적용됨을 알 수 있다.

Fig. 7. Comparison of results before (left) and after (after) correction of the pitch value

비행계기, Google Earth와 함께 모션 플랫폼을 동작시키기 위하여 시리얼 통신을 이용하여 아두이노 디바이스에 동작 값을 전송해야 한다. 각 시스템마다 설정되어 시리얼 포트가 다를 수 있기 때문에 엔진이 시작되면 해당 시리얼 포트를 별도로 지정하도록 한다. 시리얼 통신을 위한 클래스 객체의 핵심적인 메소드는 Fig. 7과 같다.

```
int CSerialComm::connect(char* portNum)
{
        if (!serial.OpenPort(portNum))
                return RETURN_FAIL;
        serial.ConfigurePort(CBR_9600, 8, FALSE,
        NOPARITY, ONESTOPBIT);
        serial.SetCommunicationTimeouts(0,0,0,0,0);
        return RETURN_SUCCESS;
}

int CSerialComm::sendCommand(char *buffer,
                                unsigned int buf_size)
{
        if (serial.WriteByte(buffer, buf_size))
                return RETURN_SUCCESS;
        else
                return RETURN_FAIL;
}

void CSerialComm::disconnect()
{
        serial.ClosePort();
}
```

Fig. 8. Methods for serial communication

connect 메소드는 해당 포트를 열고 통신 속도 등의 포트 기본값과 Timeout 값을 설정한다. sendCommand 메소드는 데이터를 전송하는 메소드이며 WriteByte 메소드을 이용하며 해당 메소드의 코드는 Fig. 8과 같다.

```
bool CSerialPort::WriteByte(char *buffer,
                                unsigned int buf_size)
{
  m_iBytesWritten = 0;
  std::cout << buffer << " ";
  if (WriteFile(m_hComm, (void*)buffer, buf_size,
                &m_iBytesWritten, NULL) == 0)
    return false;
  else
    return true;
}
```

Fig. 9. WriteByte Method

이렇게 구현된 클래스를 이용하여 해당 엔진에서는 Fig. 9의 코드와 같이 해당 포트를 설정하고 데이터를 모션 플랫폼의 아두이노에게 전송한다. 전송되는 정보는 Roll과 Pitch 값이다. 항적 정보의 값은 음수의 범위에서 양수의 범위까지 넓지만 실

제 모션 플랫폼에서 사용하는 값은 서보모터를 동작시키는 각도 값이기 때문에 0 ~ 180 범위의 값이다. 이를 위하여 변환 매핑 함수를 이용하여 값을 변환하여 전송한다.

```
char *buffer = new char[20];
CSerialComm serialComm;
long roll_ser, pitch_ser;
char *serial = new char[10];

std::cout << "Serial Port : " << endl;
cin >> serial;
if (!serialComm.connect(serial))
{
   std::cout << "Serial connect faliled" << endl;
   return -1;
}
else
   std::cout << "Serial Connect Successed" << endl;
:
:
roll_ser = (long)map(roll, -100.0, 100.0, 0.0, 180.0);
pitch_ser = (long)map(pitch, -40.0, 40.0, 0.0, 180.0);
sprintf(buffer, "%03d₩n", (int)roll_ser);
if (!serialComm.sendCommand(buffer, sizeof(buffer)))
  std::cout << "send command failed" << endl;
sprintf(buffer, "%03d₩n", (int)pitch_ser);
if (!serialComm.sendCommand(buffer, sizeof(buffer)))
  std::cout << "send command failed" << endl;
```

Fig. 10. Motion information transmission using serial communication

모션 플랫폼의 서보 모터를 동작하기 위해서는 전송된 Roll 값과 Pitch값 외에 Yaw 값이 필요하다. 항적 정보에서 Yaw 값을 받을 수 있지만 동작의 단순성을 위하여 두 값만 사용한다. 먼저 6개 서버모터를 초기화 시키고 loop 함수에서 주기적으로 시리얼 전송된 두 값을 읽어 각 서보 모터에 적용하기 위한 각도 값을 계산한다.

Ⅴ. Conclusions

본 연구에서는 실제 항적정보를 기준하여 6축 모션 플랫폼을 구현하여 동작을 실효성을 검증하였고 계기정보와 Google Earth를 연동하여 시각적인 표현의 사실성을 확인하였다. 6축 모터 제어를 위한 수학적 모델을 기반 한 동작코드를 아두이노 기반으로 구현하였고 OpenGL과 C++언어를 이용하여 계기 시뮬레이터를 구성하였다. 또한 Google Earth을 연동시켜 오픈소스 기반의 시각화 결과물을 완성하였다. 본 연구의 결과는 항공관련 교과목을 배우는 학생들의 교육 자료 및 항공기 비행을 이해하고 가상 시뮬레이션을 개발하는데 활용될 수 있을 것으로 사료된다. 향후, 본 결과물을 바탕으로 실제 계기에 따른 동작을 연동하여 실질적인 시뮬레이션을 제작할 예정이다.

REFERENCES

[1] Pontzer, A.E., Lower, M.D., Miller, J.R., “Unique aspects of flight testing unmanned aircraft systems,” In: Flight Test Technique Series, Vol. 27, pp. 1-78, 2010.

[2] Park, S., Park, M., “3D Visualization for Flight Situational Awareness using Google Earth,” Journal of The Korea Society of Computer and Information, Vol. 15, No. 12, pp. 181-188 (Dec 2010)

[3] Park, M.C., Ha, S.W., “The visualization tool of the open-source based for flight waypoint tracking,” In: Kim, T.h., Adeli, H., Robles, R.J., Balitanas, M. (eds.) Ubiquitous Computing and Multimedia Applications, Communicatio ns in Computer and Information Science, Vol. 151, pp. 153-161. Springer Berlin Heidelberg , 2011.

[4] TANG Yong, HU Minghua, WU Honggang, "3D Simulation of A-SMGCS Surface Movement based on FlightGear", JDCTA: International Journal of Digital Content Technology and its Applications, Vol. 6, No. 20, pp. 192-200, 2012.

[5] Arduino, http://arduino.cc/

[6] D-B Yoon, K-Y Lee, S-G Han, Y-H Kim, S-D Lee, “A Study on Flight Stabilization of Drones by Gyro Sensor and PID Control,” The Journal of the Korea Institute of Electronic Communication Sciences, Vol. 12, No. 4, pp. 591-598, 2017.

[7] Hyunsoo Ha, ByungYeon Hwang, “Machine Learning Model of Gyro Sensor Data for Drone Flight Control,” Journal of Korea Multimedia Society Vol. 20, No. 6, pp. 927-934, 2017.

[8] Park, M.C., Park, S.G., “The Implementation of Visualization for Ski Jump Using OpenGL,” Journal of the Korea Society of Computer and Information, Vol. 16, No. 11, pp. 137-143, 2011.

[9] Moin Uddin Atique, Atiqur Rahman, “AhadInverse Kinematics solution for a 3DOF robotic structure using Denavit-Hartenberg Convention,” 2014 International Conference on Informatics, Electronics & Vision (ICIEV), pp. 1-5, 2014.

[10] Dasol Lee and David Hyunchul Shim, “Design and Validation of Low-cost Flight Control Computer for Multi-rotor UAVs,” Journal of the Korean society for

aeronautical & space sciences, Vol. 45, No. 5, pp. 401-408, 2017.
[11] World Street Map, http://www.arcgis.com
[12] TANG Yong, WU Honggang, LIU Pengfei, SI Quan, "Real-time 3D flight track and flight simulation based on Google Earth," JDCTA: International Journal of Digital Content Technology and its Applications, Vol. 6, No. 19, pp. 385-392, 2012.

Authors

Myeong-Chul Park received a B.S. degree in Computer Science from Korea National Open University in 1999, a M.S. and Ph.D. degrees in Computer Science from GyeongSang National University in 2002, 2007. He is currently a Professor in the Department of Avionics Engineering, KyungWoon University. He is interested in Visualization, Simulation, Education of Software, Virtual Reality, and Parallel Programming.

Duk-Kyu Choi received a B.S. degree in Electronic Engineering from Kyungpook National Univeristy in 1990, a M.S. and Ph.D. degrees in Electronic Engineering from Kyungpook National Univeristy in 1993, 1997. He is currently a Professor in the Department of Avionics Engineering, KyungWoon University. He is interested in Embedded system, Digital video technology, Mobile Telecommunication.