

# Forward Kinematics

-Computer Problem Set 2-

2018. 05. 23.

미래융합기술학과

20187087 조은기

## 1. Introduction

- ① 프로젝트 목적
- ② 사용 Tool 및 라이브러리

## 2. Algorithm

- ① Manipulator 설계
- ② 동역학 수식 유도

## 3. Demo

## 4. Source Analysis

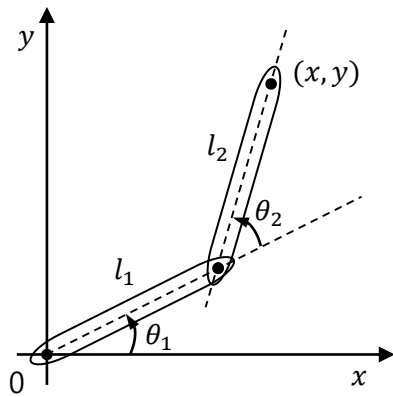
## 5. Impression



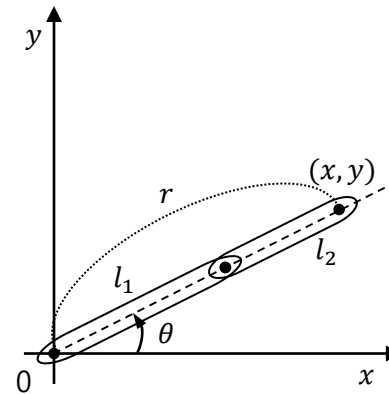
# 1. Introduction

## ① 프로젝트 목적

- 프로그래밍 언어와 그래픽 라이브러리를 사용한 Manipulator 설계
- Kinematics를 이용한 Manipulator control simulation 구현



<  $\theta_1 - \theta_2$  Planar Robot >



<  $\theta - r$  Planar Robot >

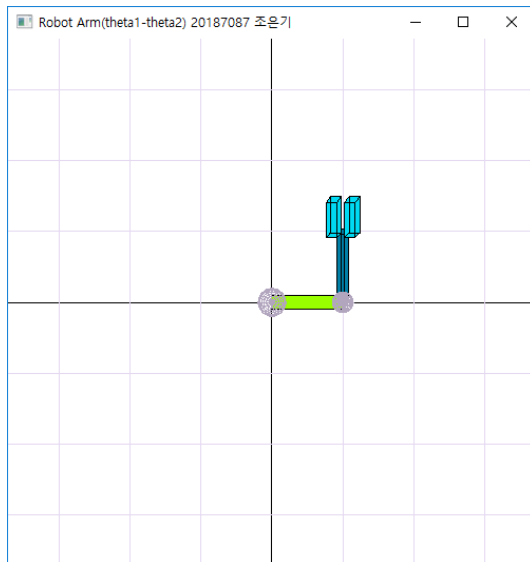
## ② 사용 Tool 및 라이브러리

- Visual studio 2013 (C 언어)
- OpenGL
- Windows API

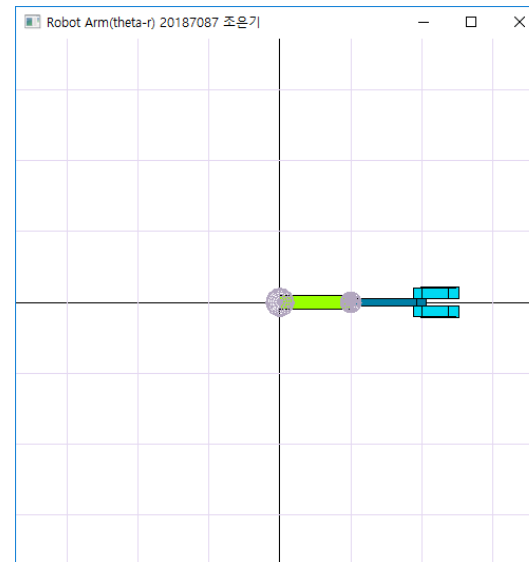
## 2. Algorithm

### ① Manipulator 설계

- Windows API와 OpenGL을 이용하여 2 DOF Manipulator 설계
- 1축 Prismatic joint, 2축 Prismatic joint로 구성된  $\theta_1 - \theta_2$  Planar Robot
- 1축 Prismatic joint, 2축 Revolute joint로 구성된  $\theta - r$  Planar Robot



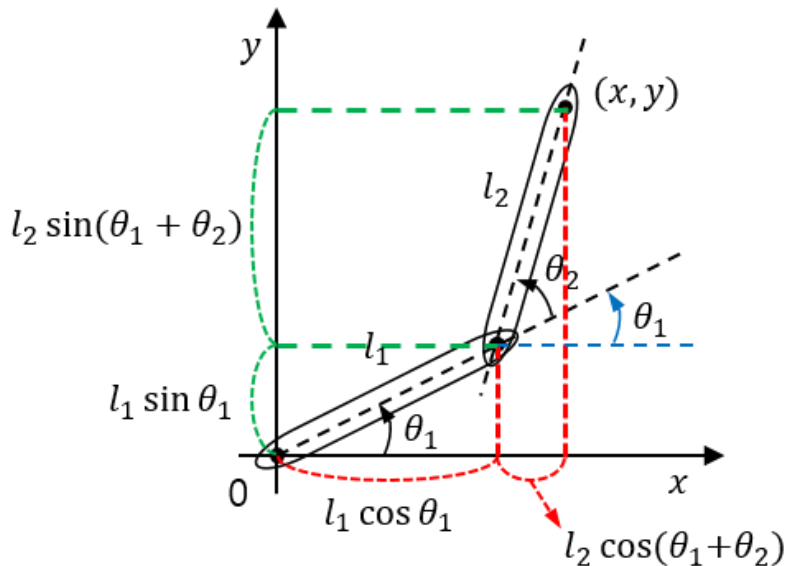
<  $\theta_1 - \theta_2$  Planar Robot >



<  $\theta - r$  Planar Robot >

## 2. Algorithm

### ② 동역학 수식 유도 ( $\theta_1 - \theta_2$ Planar Robot)



$$P_x = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2)$$

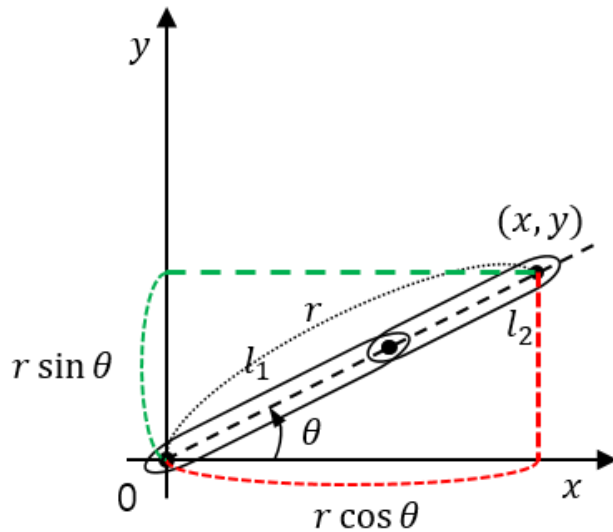
$$P_y = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2)$$

$$\begin{bmatrix} \mathbf{R} & \begin{matrix} P_{x_0} \\ P_{y_0} \\ 0 \\ 1 \end{matrix} \end{bmatrix} = \begin{bmatrix} \cos \theta_1 & -\sin \theta_1 & 0 & 0 \\ \sin \theta_1 & \cos \theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & l_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta_2 & -\sin \theta_2 & 0 & 0 \\ \sin \theta_2 & \cos \theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) & 0 & l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) & 0 & l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## 2. Algorithm

### ② 동역학 수식 유도 ( $\theta - r$ Planar Robot)



$$P_x = r \cos \theta$$

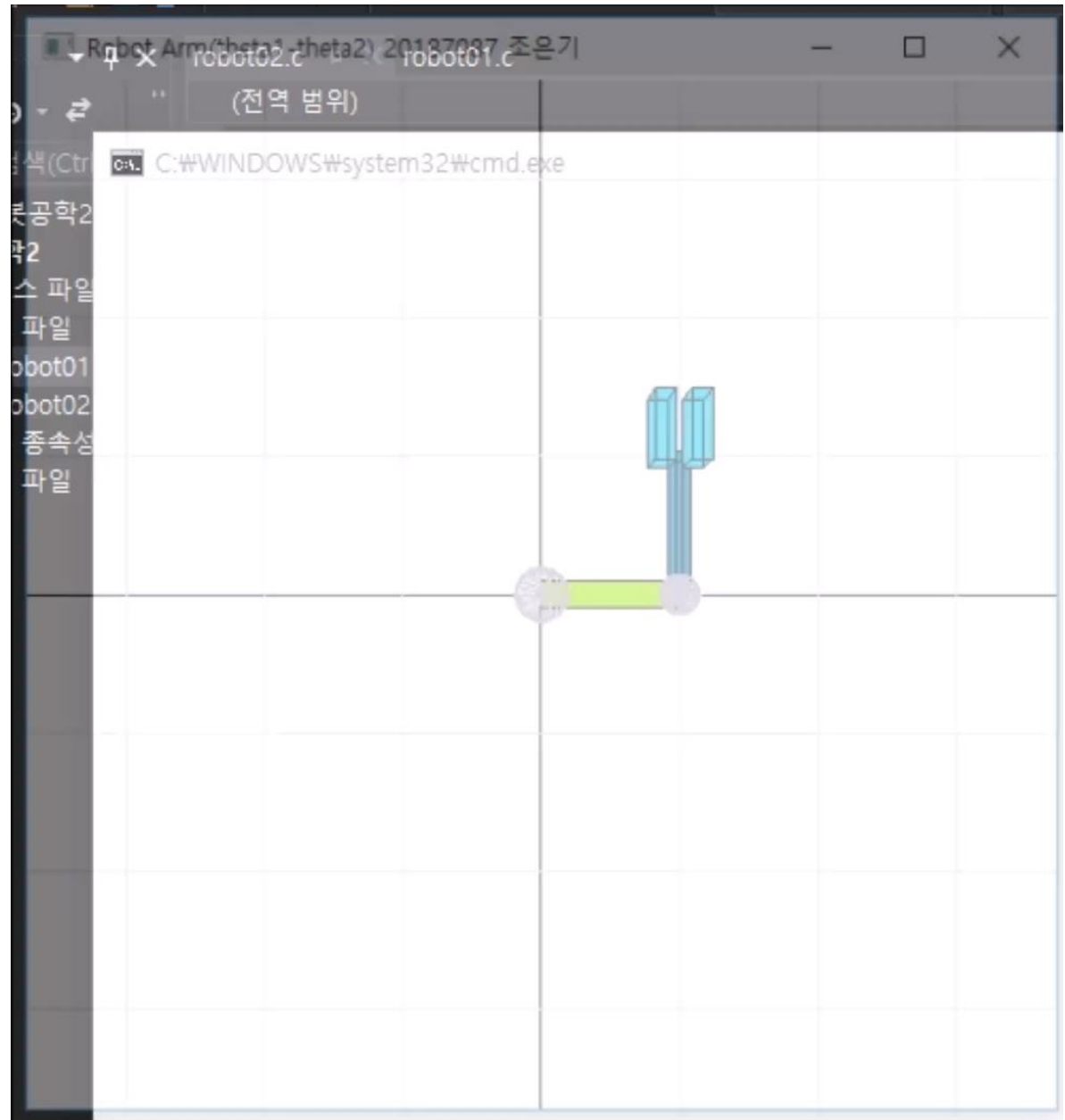
$$P_y = r \sin \theta$$

$$\begin{bmatrix} \begin{matrix} \mathbf{R} \end{matrix} & \begin{matrix} P_{x_0} \\ P_{y_0} \\ 0 \\ 1 \end{matrix} \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & r \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & r \cos \theta \\ \sin \theta & \cos \theta & 0 & r \sin \theta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

### 3. Demo

①  $\theta_1 - \theta_2$

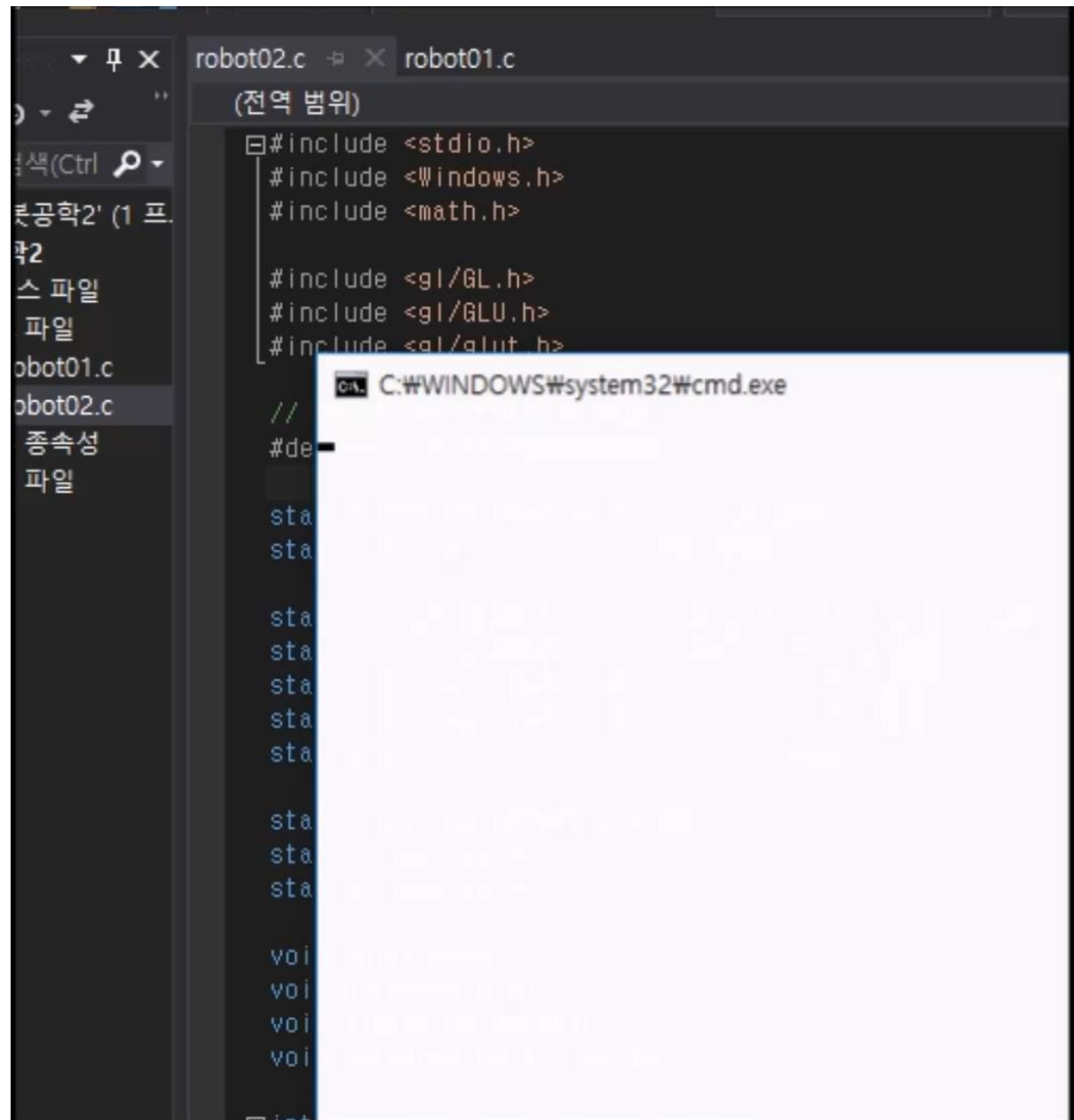
**Planar Robot**



### 3. Demo

②  $\theta - r$

Planar Robot



The screenshot shows a C++ IDE with two files open: robot02.c and robot01.c. The robot01.c file is active and displays the following code:

```
(전역 범위)
#include <stdio.h>
#include <Windows.h>
#include <math.h>

#include <gl/GL.h>
#include <gl/GLU.h>
#include <gl/glut.h>

//
#define ...

static ...
static ...

static ...
static ...
static ...
static ...
static ...
static ...

static ...
static ...
static ...

void ...
void ...
void ...
void ...
```

A terminal window is open in the foreground, showing the command prompt path: C:\WINDOWS\system32\cmd.exe.



# 4. Source Analysis

## ① $\theta_1 - \theta_2$ Planar Robot

```
#define PI 3.1415926535897

static GLfloat theta1 = 0; // 세타 값1 설정
static GLfloat theta2 = 0; // 세타 값2 설정

static float x[201];      // 궤도 x[201]로 설정
static float y[201];      // 궤도 y[201]로 설정
static GLfloat time = 0;   // 시간 설정
static int i, j = 0;       // 궤도

static int refreshMills = 30;

static float f1 = 1;
static float f2 = 0.5;
static float l = 1;
```

$$f_1 = 1 \text{ Hz} \quad f_2 = 0.5 \text{ Hz}$$

$$L_1 = L_2 = 1.0 \text{ m}$$

$$\theta_1 = 2\pi f_1 t$$

$$\theta_2 = 2\pi f_2 t + \pi/2$$

```
theta1 = 2 * PI*f1*time;      // 로봇팔 공식
theta2 = 2 * PI*f2*time + 1.57; // 90도 꺾인 것을 표현해주기위해 1.57radian을 더해줌
```

```
x[j] = ((l*cos(theta1)) + (l*cos(theta1 + theta2))); // 궤적 공식
y[j] = ((l*sin(theta1)) + (l*sin(theta1 + theta2)));
```

$$P_x = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2)$$

$$P_y = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2)$$

```
for (i = 0; i < j; i += 2) { // 궤적
    glVertex2f(x[i], y[i]);
}
```

```
if (time < 2.00 - 0.01) { // 시간이 2초가 되면 멈출
    j++;
    time += 0.01;
}
```

*total time = 2 second*

*sequence = 0.01*

# 4. Source Analysis

## ② $\theta - r$ Planar Robot

```
#define PI 3.1415926535897

static GLfloat theta = 0; //  $\theta$  값 설정
static GLfloat r = 2; // r 값 설정

static float x[201]; // 궤도 x[201]로 설정
static float y[201]; // 궤도 y[201]로 설정
static GLfloat time1 = 0; // 시간1 설정
static GLfloat time2 = 0; // 시간2 설정
static int i, j = 0; // 궤도
```

```
static int refreshMills = 30;
static float f1 = 1;
static float f2 = 1;
```

```
theta = 2 * PI * f1 * time1; // 로봇팔 공식
r = 2.0 - f2 * time2;
```

```
x[j] = r * cos(theta); // 궤적 공식
y[j] = r * sin(theta);
```

```
for (i = 0; i < j; i += 2) { // 궤적
    glVertex2f(x[i], y[i]);
}
```

```
if (time1 < 2.00 - 0.01) { // 시간이 2초가 되면 멈출
    j++;
    time1 += 0.01;
```

```
    if (time1 < 1.00 - 0.01)
        time2 += 0.01;
    else if (time1 >= 1.00 - 0.01 && time1 < 2.00 - 0.01)
        time2 -= 0.01;
}
```

$$f_1 = 1 \text{ Hz} \quad f_2 = 0.5 \text{ Hz}$$

$$\theta = 2\pi f_1 t_1$$
$$r = 2.0\text{m} - f_2 t_2$$

$$P_x = r \cos \theta$$

$$P_y = r \sin \theta$$

$$\text{total time} = t_1 = 2 \text{ second}$$

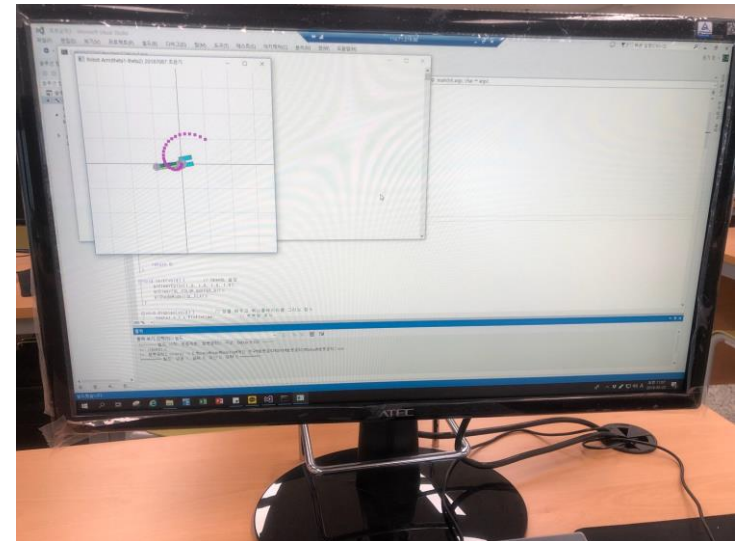
$$\text{sequence} = 0.01$$

$$t_2 = 0 \rightarrow 1 \quad (t_1 = 0 \sim 1)$$

$$t_2 = 1 \rightarrow 0 \quad (t_1 = 1 \sim 2)$$

## 5. Impression

Manipulator simulator를 설계하고 Forward kinematics를 적용시켜 봄으로써 Joint들의 각도나 길이 조정으로 End-effector의 위치와 방향을 결정하는 Kinematics에 대해 이해할 수 있었고 과제에서는 2 DOF만을 다뤘지만 더 많은 자유도에 대한 Kinematics를 생각해 볼 수 있었습니다.



# Q & A

감사합니다.