

# E조

엄태경

김민은

장병우

심상원

# 개요

---

1.Yolo

2.Arduino

3.Kinematics

Yolo

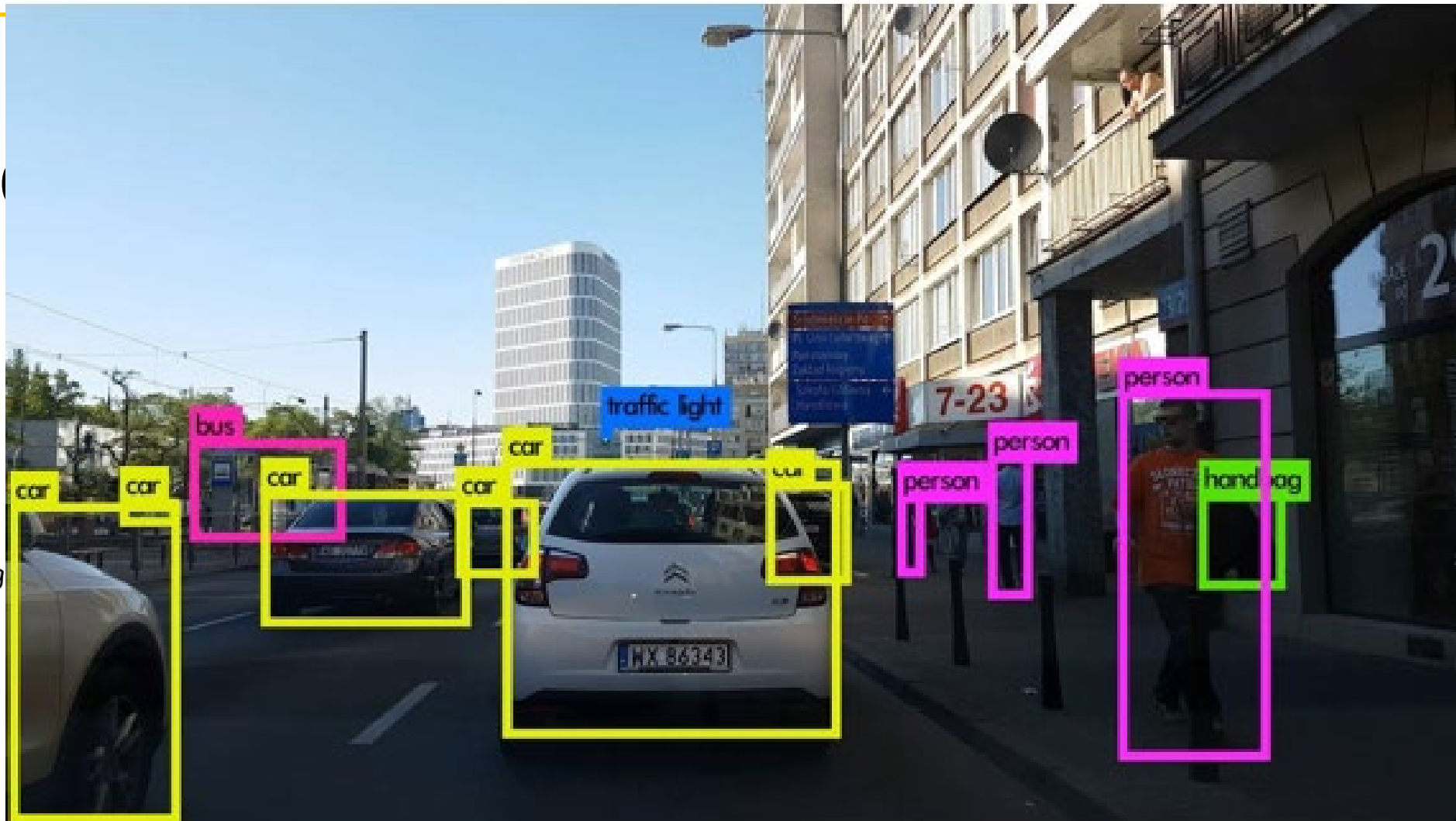
# 0. Object Detection 이란?

Object

tion

Cat

Dog



# Yolo – You Only Look Once

---

1 – stage Detector: Yolo, SSD

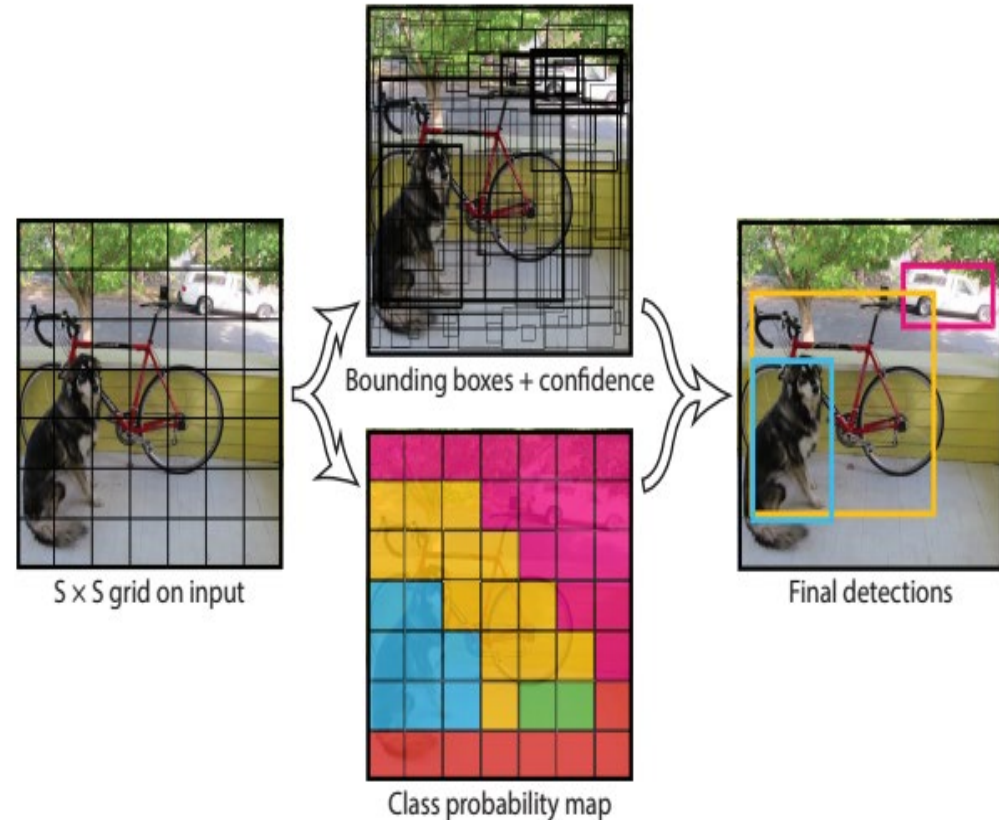
2 – stage Detector: R-CNN

# Yolo Version 1

1. 이미지를  $S \times S$  Grid Cell로 나눈다.
2. 각 Cell 마다 2개의 Bounding Box를 가진다.
3. 또한, 각 Cell 마다 Classification을 수행한다.

## 문제점

1. 각 Cell마다 하나의 클래스만 예측
2. 작은 물체에 대한 인식률이 낮음



# Yolo Ver 2, 3

## Yolo ver 2

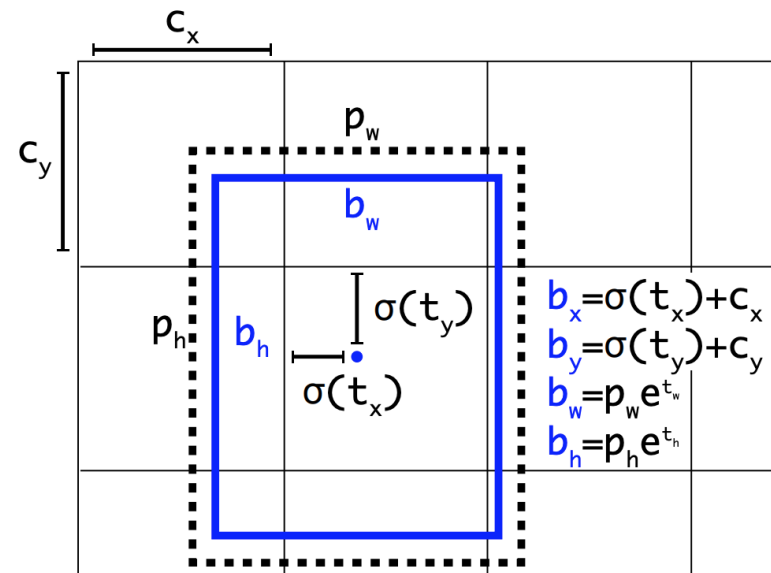
1. 각 Cell 마다 5개의 Anchor Box를 둔다.
2. 각 Anchor Box 마다 Classification 진행

## Yolo ver 3

작은 사이즈의 Feature map – Big Object 유리  
큰 사이즈의 Feature map – Small Object 유리

=> Feature map을 3개의 Scale 로 나눔

Anchor box 각 Scale 별로 3개씩 총 9개



Anchor Box란?

-입력 영상에 대해서 객체가  
있을 법한 곳에 미리 설정한 박스

# Yolo Ver 1,2,3 비교

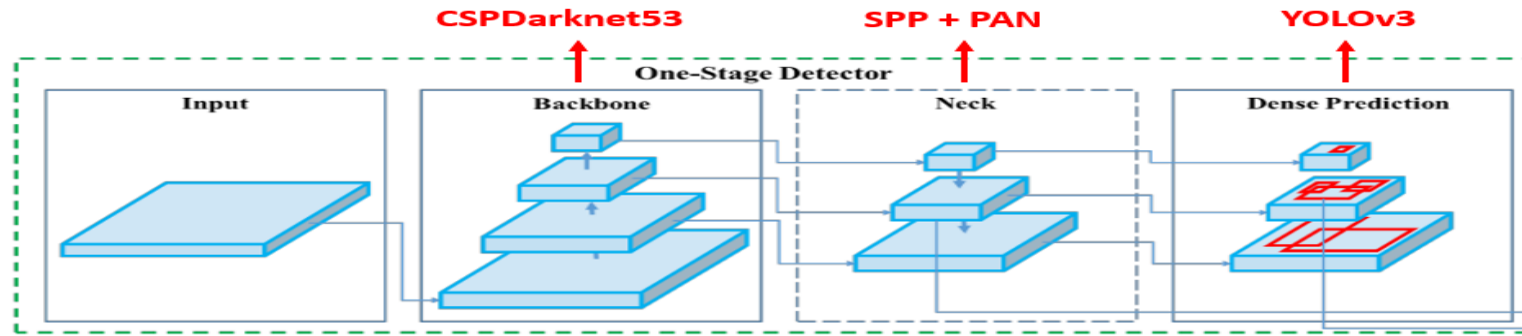
---

Version	Input Image Size	Prediction Grid Cells	Boxes per cell	No. of BBox
1	448X448	7X7	2	98
2	416X416	13X13	5	845
3	416X416	52X52, 26X26, 13X13	3	10,647

= (52X52X3) + (26X26X3) + (13X13X3)



# Yolo Ver 4



$$\text{YOLOv4} = \text{YOLOv3} + \text{CSPDarknet53} + \text{SPP} + \text{PAN} + \text{BoF} + \text{BoS}$$

↓  
Path Aggregation Network

Model Architecture가 Backbone – Neck – Head 구성

1. Backbone: Image로부터 feature를 찾아낸다.
2. Neck: 찾아낸 feature를 aggregation
3. Head: Bounding Box + Classification 진행

# Yolo Ver 7

---

1. Model re-parameterization

2. Dynamic label assignment

최신 기술을 이용하여 성능을 극대화함

# Yolo Ver 7 성능

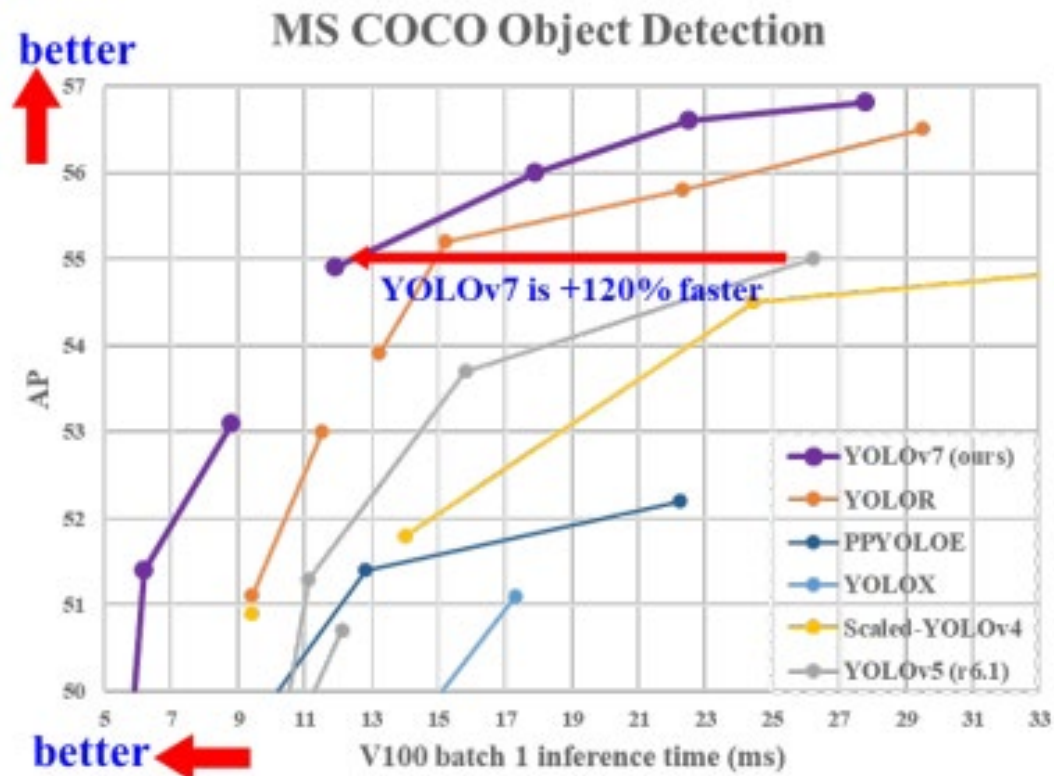
Version 4 와 비교하여

1. 75% 적은 Parameters
2. 36% 적은 Comtutational Time
3. 1.5배 높은 AP

현존하는

30FPS 이상 Real Time Object detector 중

가장 정확하고, 빠르다!!!



# 기술 스택

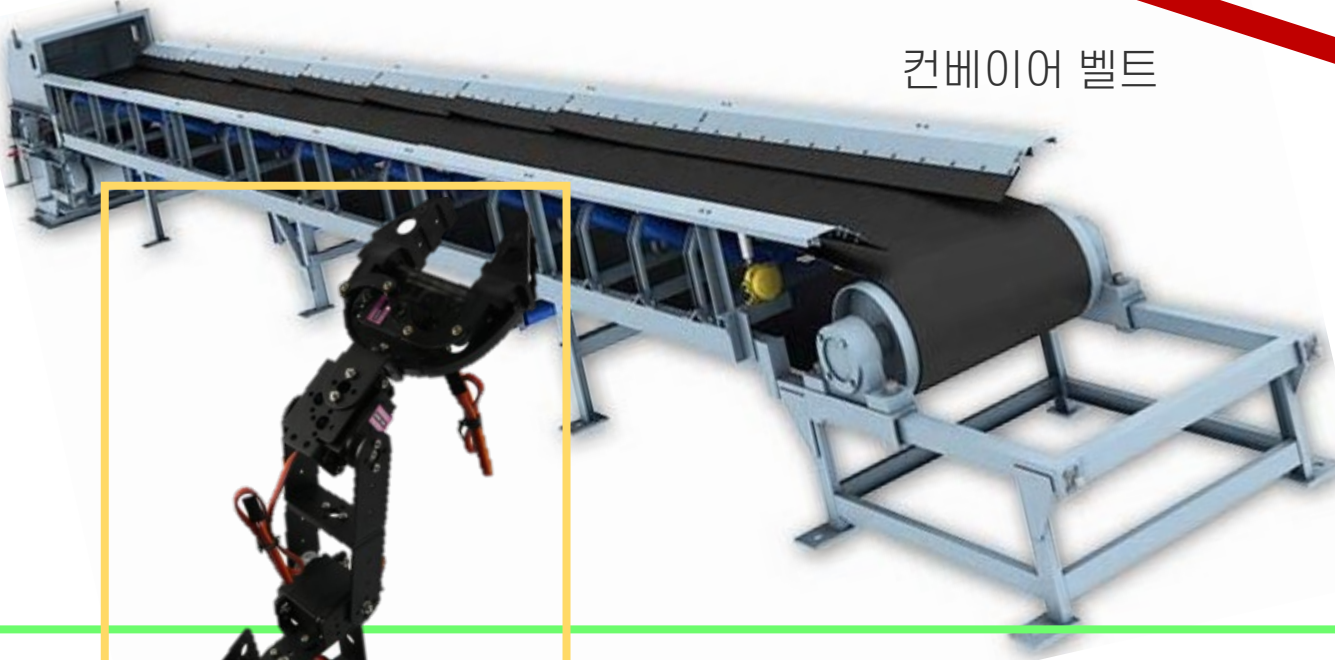
아두이노, Kinematics, YOLO

카메라 모듈



1. 영상 정보 실시간 송출

컨베이어 벨트



2. PET 검출 및 그리퍼 각도 계산



PC

3. 각도 전송



그리퍼

4. 그리핑

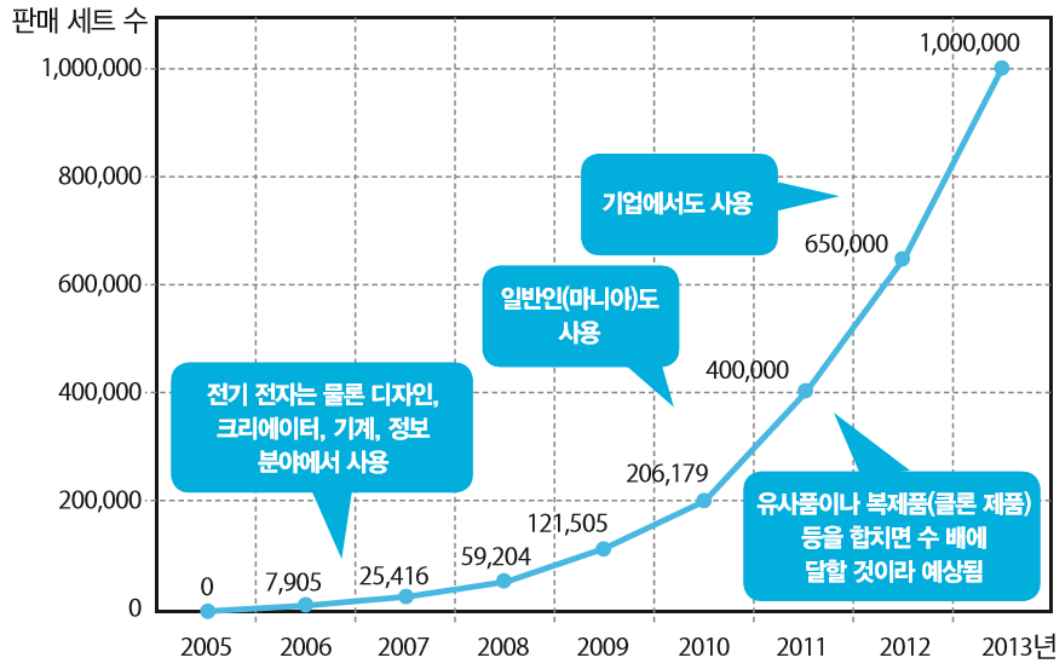
# 아두이노

## 아두이노의 탄생과 배경

2005년 이탈리아 Ivera 마을 IDE 대학교 부교수 **마시모 반지**

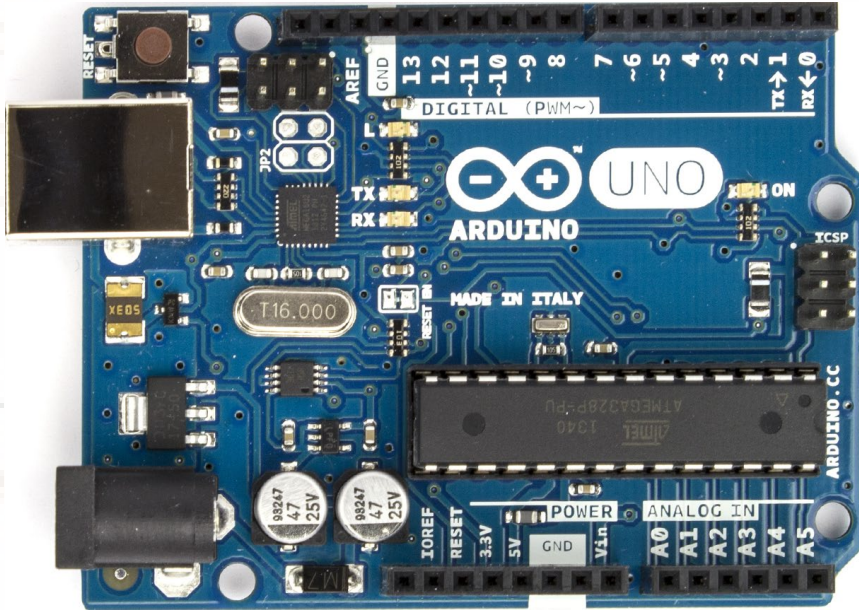
전기전자 전공생들이 손쉽게 공부할 수 있도록 값싼 교재용 **Micro Computer(마이콤)** 목적

아두이노가 오픈 소스 하드웨어로 개발되었기에 보급이 확대되어 오늘날 **학생과 일반인은 물론 여러 기업의 기술자도 사용**

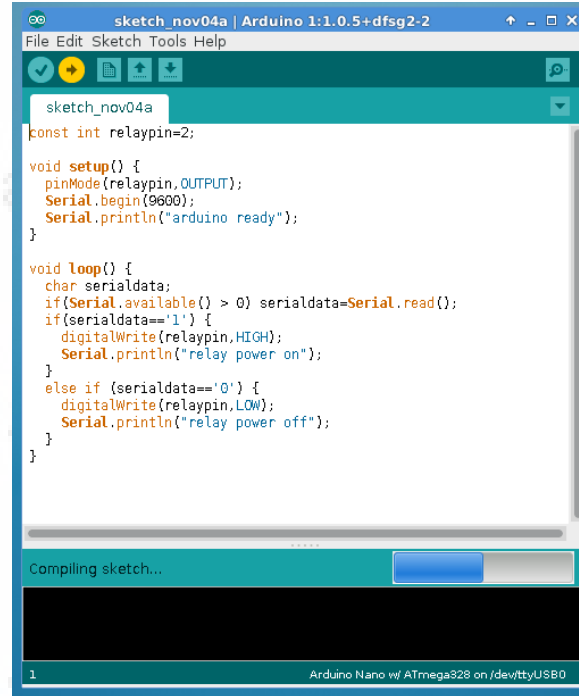


# 아두이노

아두이노 통합 개발 환경



마이크로 컨트롤러 보드

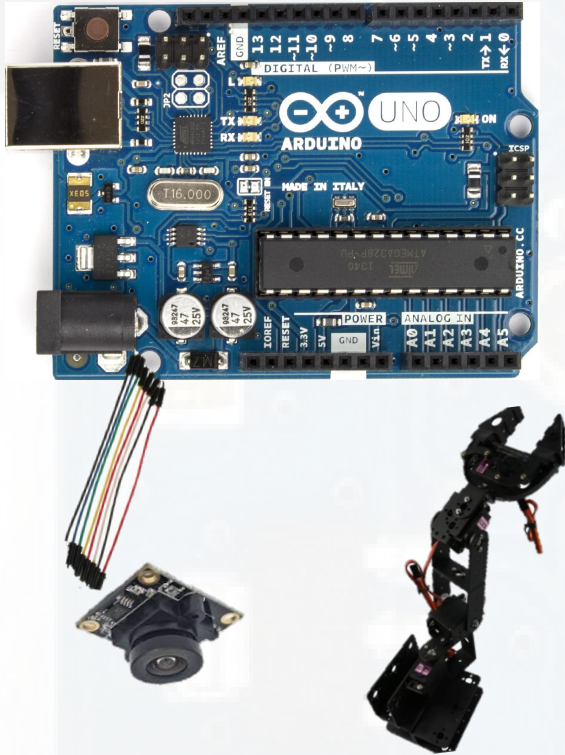


IDE



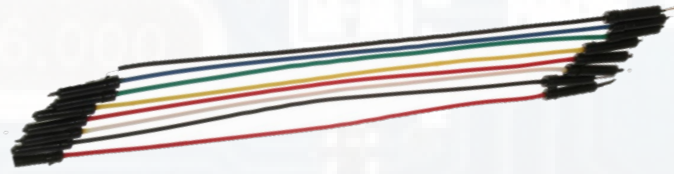
# 아두이노

아두이노 통합 개발 환경

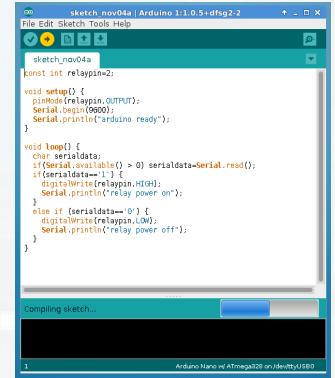


① 아두이노 + 전자 부품 조립

④ 바이너리 파일 업로드



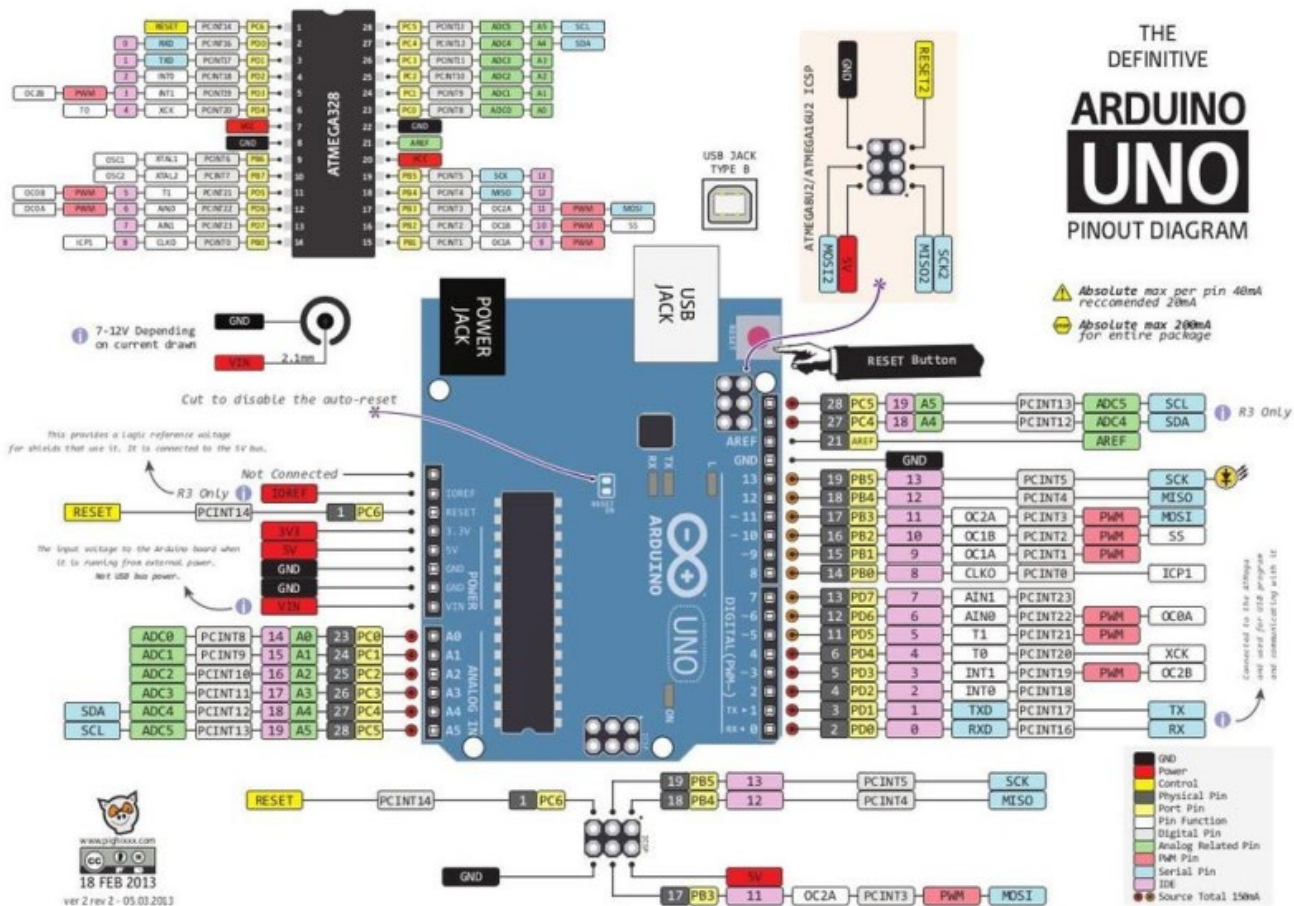
③ 스케치 컴파일



② IDE에서 스케치 작성 및 디버깅

# 아두이노

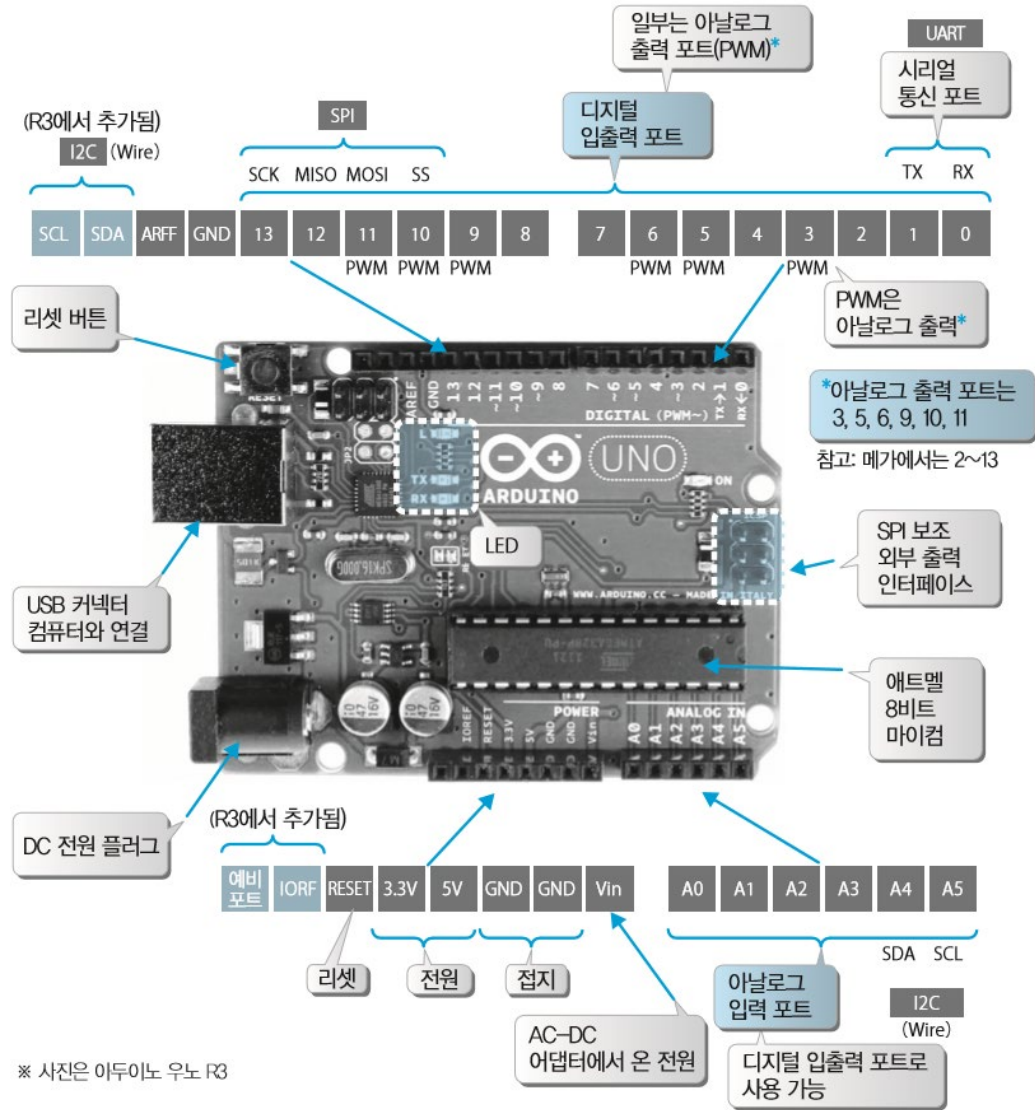
## Arduino Uno R3 Pinmap





# 아두이노

## Arduino Uno R3 Pinmap



※ 사진은 아두이노 우노 R3

1. USB 전원 커넥터

2. 외부 전원 커넥터

3. LED 3개

4. 아날로그 입력 포트

5. 아날로그 출력 포트

6. 디지털 입출력 포트

7. 전원과 접지 포트

8. I2C와 SPI 포트

9. UART(시리얼 통신) 포트

<전원>

<아날로그 입출력>

<디지털 입출력>

<통신>

# 아두이노

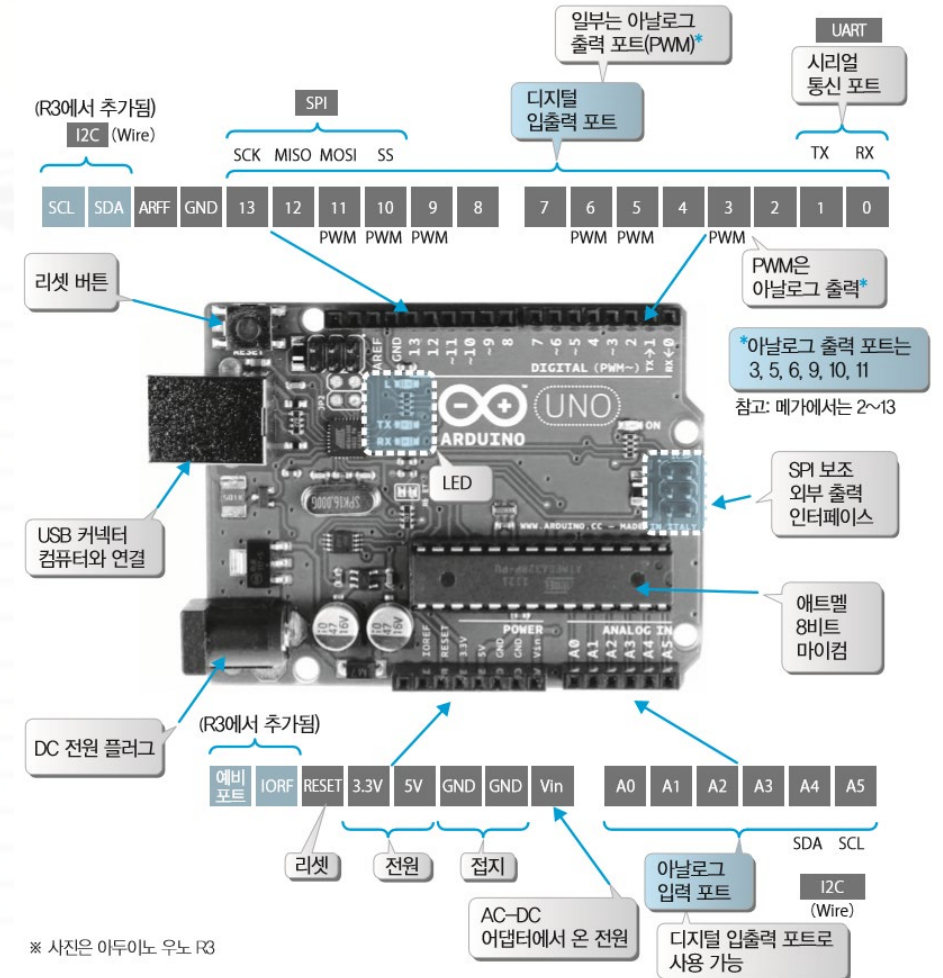
## Arduino Uno R3 Pinmap - 전원

### USB 전원 커넥터

- 컴퓨터의 USB나 5V USB 외부 전원을 연결
- 컴퓨터와 연결하면 전원 공급이 되고, 시리얼 통신으로 컴퓨터에서 아두이노로 프로그램을 업로드하거나 데이터를 주고받을 수 있음

### 외부 전원 커넥터

- 전압이 7~12V인 외부 직류 전원을 연결
- 컴퓨터 연결과 동시에 사용할 수 있음(권장하는 전압은 9V 정도)



# 아두이노

## Arduino Uno R3 Pinmap - 아날로그

### LED 3개

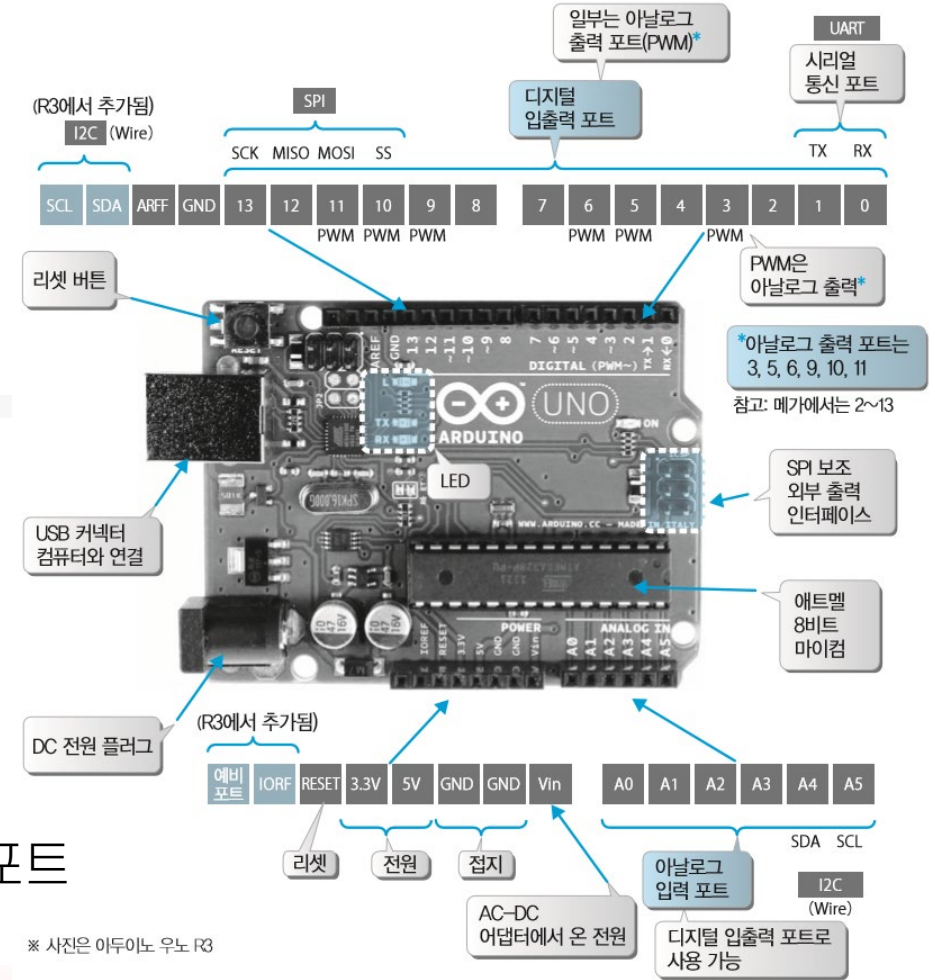
- L, TX, RX에 해당하는 LED 3개가 아두이노 로고 옆에 배치되어 있음
- 시리얼 통신을 할 때 켜지거나 꺼짐

### 아날로그 입력 포트

- A0에서 A5까지 아날로그 입력 포트 6개가 있음
- 이 포트는 디지털 입출력 포트 D14~D19로도 사용할 수 있음

### 아날로그 출력 포트

- 실제로는 PWM(펄스 폭 변조)을 사용해서 아날로그 신호를 출력하는 포트
- 포트 번호는 D3, D5, D6, D9, D10, D11



# 아두이노

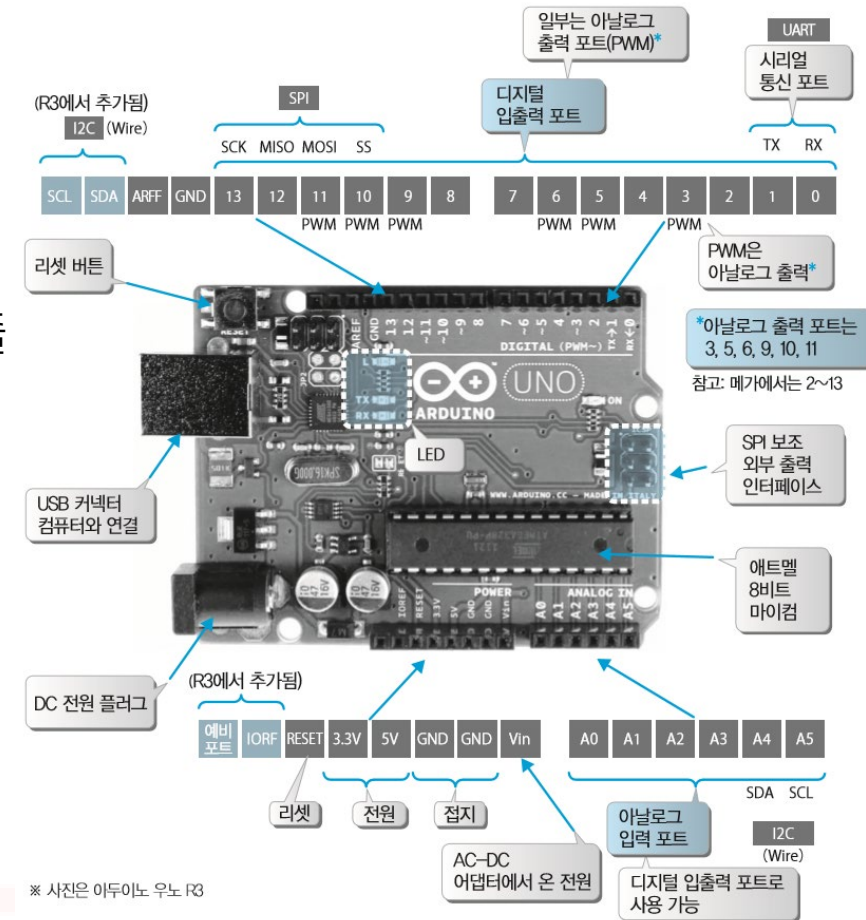
## Arduino Uno R3 Pinmap – 디지털

### 디지털 입출력 포트

- D0에서 D13까지는 디지털 입출력 포트
- 아날로그 입력 포트인 A0~A5도 디지털 입출력 포트 D14~D19로 사용 가능
- 이 포트 중 D0와 D1은 하드웨어 시리얼 통신을 할 수 있고, 빠른 속도로 입출력을 할 수 있음

### 전원과 접지 포트

- 3.3V 또는 5V 전원과 접지 3개가 있음
- 이외에 Vin 포트를 사용하면 외부 전원을 바로 사용할 수 있음
- DC(직류) 전원 커넥터와 같은 방법으로 7~12V 직류 전원을 꽂아 사용



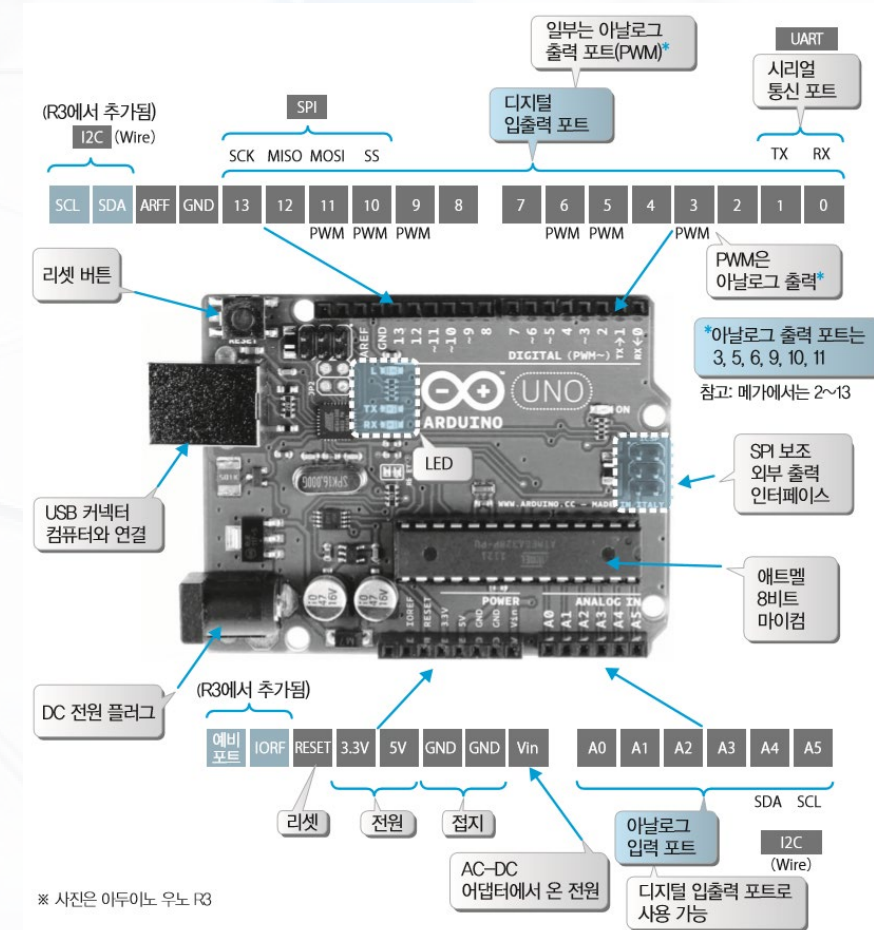


# 아두이노

## Arduino Uno R3 Pinmap – 통신

### I2C와 SPI 포트

- 동기식 버스 시리얼 통신을 할 수 있는 포트
- 전송 가능한 거리는 짧지만 빠른 속도로 통신할 수 있음
- I2C 버스는 SCL(Serial Clock) 신호선과 양방향 SDA(Serial Data) 신호선, 총 2개의 신호선(접지 미포함)으로 통신
- SPI 버스는 SCK(Serial Clock) 신호선과 단방향 SD0, SD1, 이렇게 총 3개의 신호선(접지 미포함)으로 통신



※ 사진은 아두이노 우노 R3

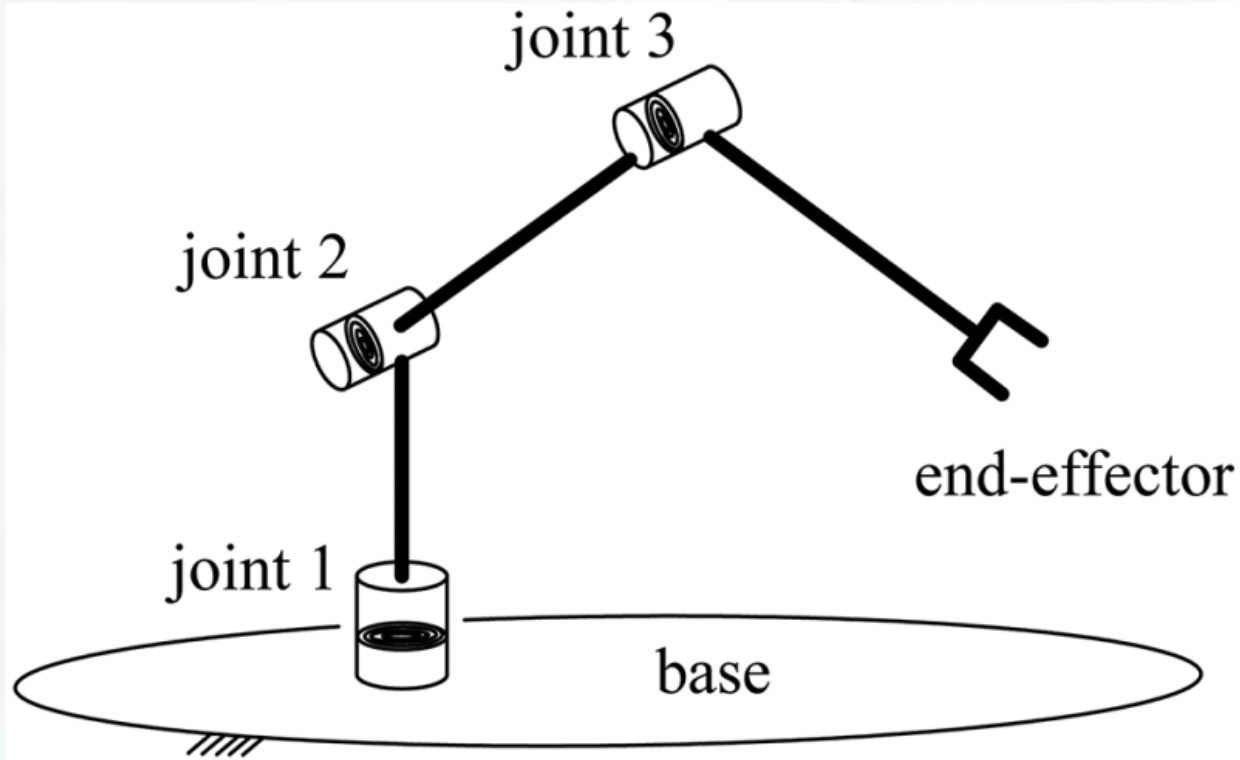
# Kinematics

## 로봇 기구학

로봇이 움직이려면 로봇의 팔 끝이 어디에 있는가를 알아야함

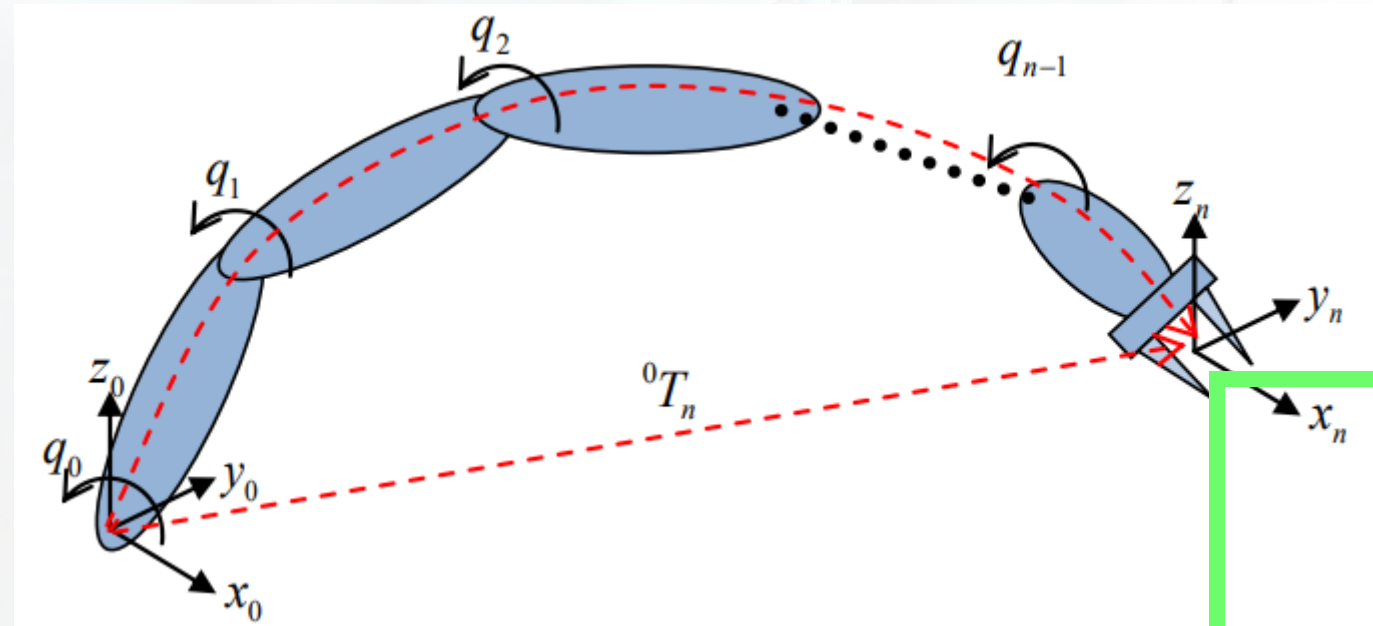
원점에서 얼마나 떨어져 있는가를 나타내는 위치와 얼마만큼 회전되어 있는가를 나타내는 자세로 정의할 수 있음

= 기구학 (로봇에서 고려되는 링크의 위치, 속도, 가속도의 관계를 다루는 학문)

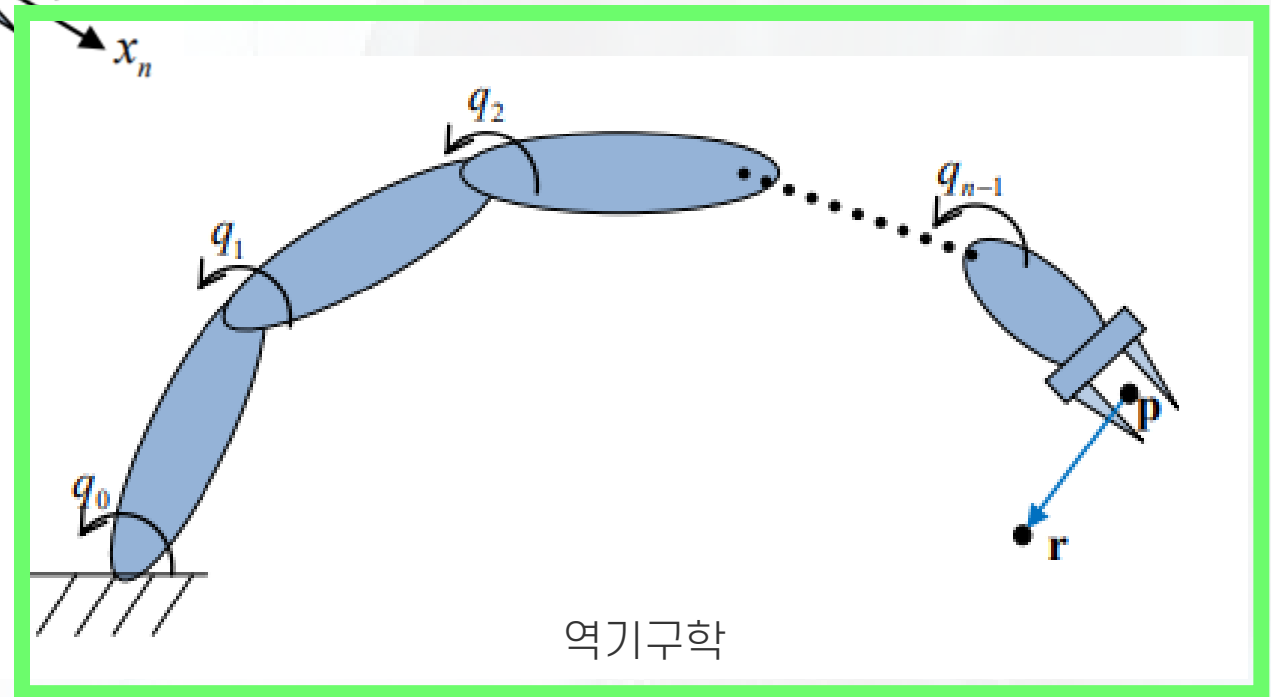


# Kinematics

역기구학과 정기구학



정기구학

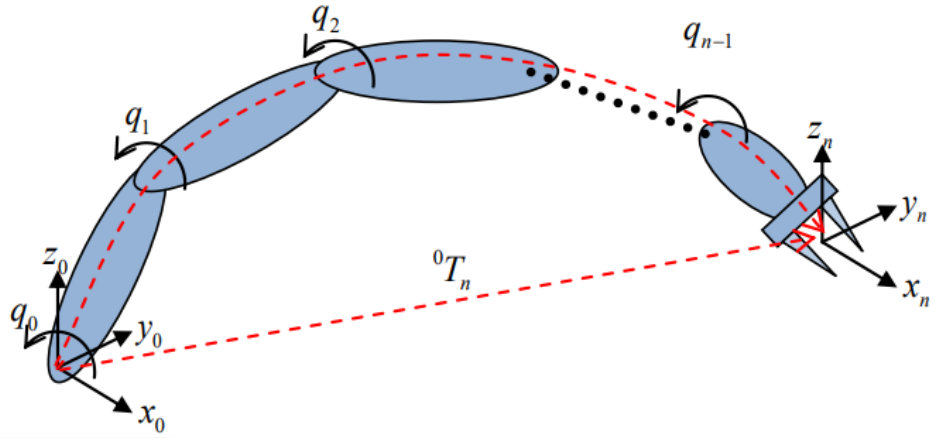


역기구학



# Kinematics

## Forward Kinematics



$${}^0\mathbf{T}_n = {}^0\mathbf{A}_1(q_0) {}^1\mathbf{A}_2(q_1) {}^2\mathbf{A}_3(q_2) \dots {}^{n-1}\mathbf{A}_n(q_{n-1}) = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

정기구학은 일련의 관절각 ( $q_0, q_1, q_2, \dots, q_{n-1}$ ) 이 주어졌을 때, 말단장치(End-effector)의 직교좌표상의 위치 ( $p, p_y, p_z$ ) 와 자세 ( $\phi, \theta, \psi$ ) 를 구하는 문제로, 역기구학에 비해 상대적으로 간단하다.

균질변환(Homogeneous Transform)을 이용하여 정기구학의 해를 구하게 된다.

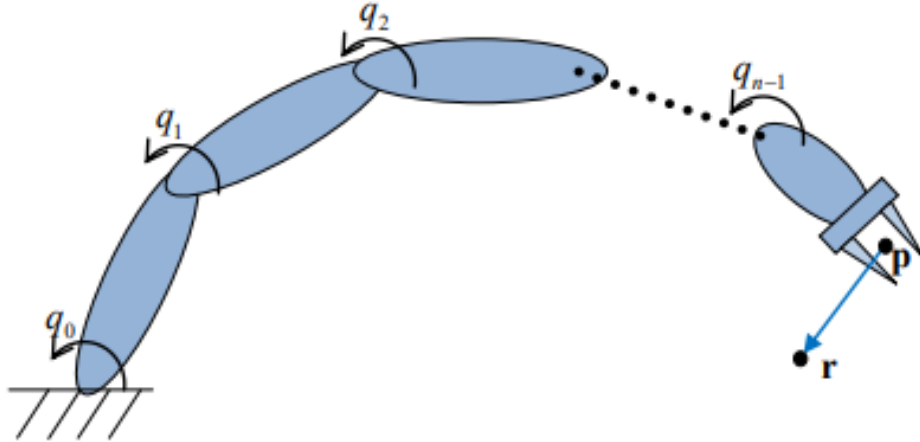
$$\psi = \tan^{-1} \left( \frac{n_y}{n_x} \right),$$

$$\theta = \tan^{-1} \left( \frac{-n_z}{\sqrt{o_z^2 + a_z^2}} \right),$$

$$\phi = \tan^{-1} \left( \frac{o_z}{a_z} \right).$$

# Kinematics

## Inverse Kinematics



<역기구학 해석 방법>

1. Moore Penrose Pseudo Inverse
2. Damped Least Square (DLS) Pseudo Inverse
3. Jacobian Transpose (JT)  $\dot{\mathbf{q}}_{JT} = \mathbf{J}^T \dot{\mathbf{p}}$
4. 가중치가 적용된 JT  $\alpha = \frac{\langle \dot{\mathbf{p}}, \mathbf{J}\mathbf{J}^T \dot{\mathbf{p}} \rangle}{\langle \mathbf{J}\mathbf{J}^T \dot{\mathbf{p}}, \mathbf{J}\mathbf{J}^T \dot{\mathbf{p}} \rangle}$

$$\begin{aligned}\mathbf{J}^{\dagger} &= \mathbf{J}^{-1} && (\text{case } m = n), \\ \mathbf{J}^{\dagger} &= \mathbf{J}^T (\mathbf{J}\mathbf{J}^T)^{-1} && (\text{case } m < n), \\ \mathbf{J}^{\dagger} &= (\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T && (\text{case } m > n).\end{aligned}$$

$$\begin{aligned}\mathbf{J}_{DLS}^{\dagger} &= \mathbf{J}^T (\mathbf{J}\mathbf{J}^T + \lambda \mathbf{I})^{-1} && (\text{case } m < n), \\ \mathbf{J}_{DLS}^{\dagger} &= (\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I})^{-1} \mathbf{J}^T && (\text{case } m > n).\end{aligned}$$

로봇은 관절에 의해 연결된 링크들의 조합으로 구성된다.

관절각(joint angle)  $\mathbf{q} = (q_0, q_1, q_2, \dots, q_{n-1})$

말단장치의 위치  $\mathbf{p} = (p_x, p_y, p_z)^T$

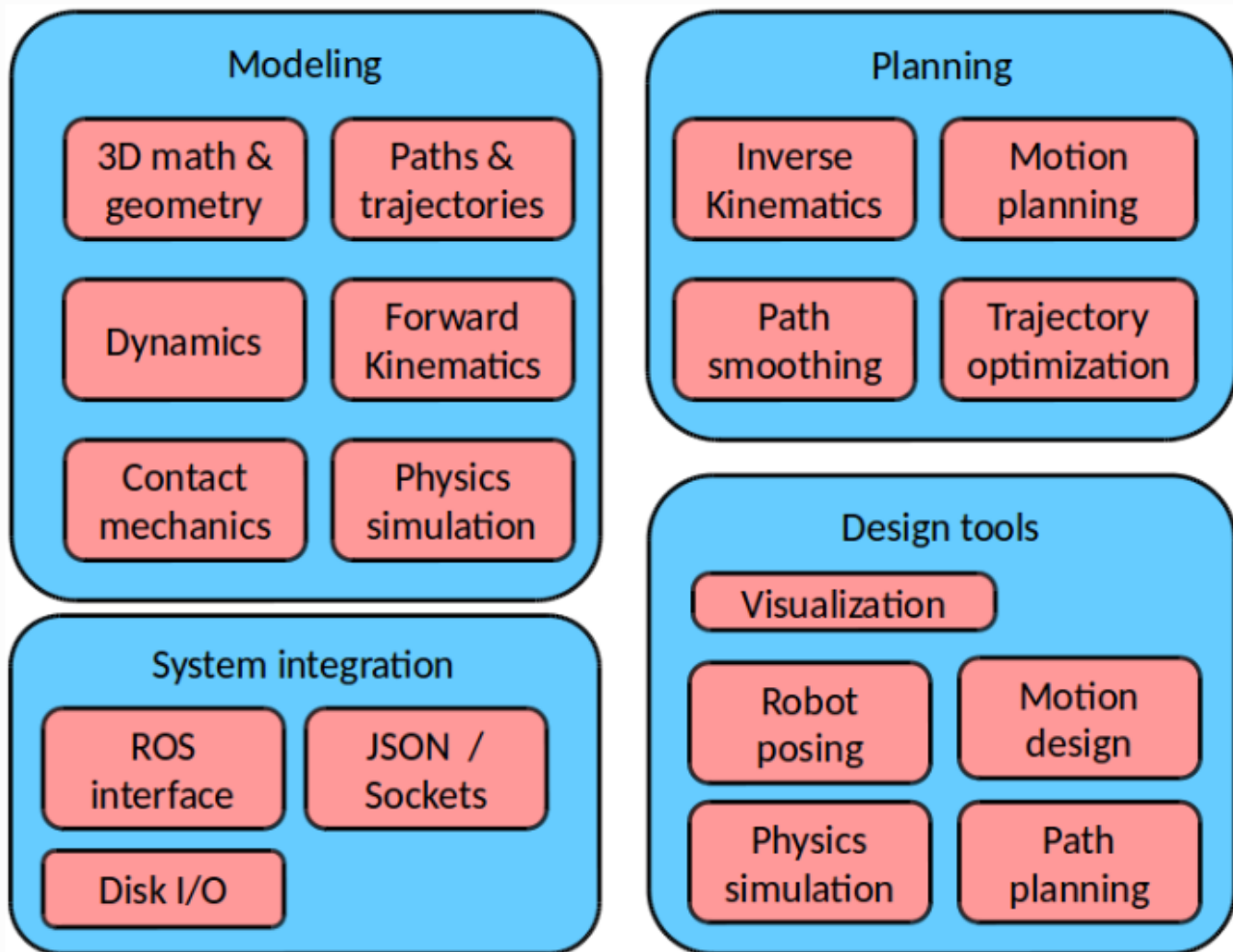
말단장치가 도달해야 할 목표 위치 역시 열 벡터  $\mathbf{r} = (r_x, r_y, r_z)^T$  로 표현 가능하며 말단장치가 목표점에 도달하기 위하여 이동해야 할 변화량을  $\vec{e} = \mathbf{r} - \mathbf{p}$  로 표현할 수 있다. 따라서 역기구학 알고리즘은  $\mathbf{r} = \mathbf{f}(\mathbf{q})$ 에서  $\mathbf{q}$ 의 값을 찾는 것이 된다.

이 식은 자코비안 행렬  $\mathbf{J}$ 에 기반한 해를 찾기 위한 반복적인 계산에 의해 풀린다.

$$\mathbf{J}(\mathbf{q}) = \frac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}}$$

# Kinematics

Open source – Klamp't Python API



## Module structure

The Klamp't Python API is organized as follows:

- `klamp't`: the main Klamp't module, and includes robot kinematics, dynamics, simulation, and geometry representations. Also includes low-level IK solving and motion planning modules.
- `klamp't.math`: basic 3D geometry.
- `klamp't.modeling`: other modeling, including IK, trajectories, Cartesian interpolation, and sub-robot indexing. Setting and getting "configurations" for many objects.
- `klamp't.plan`: motion planning for robots.
- `klamp't.sim`: more advanced simulation functionality, such as logging and custom actuator and sensor emulation.
- `klamp't.io`: Unified I/O for all types of Klamp't objects. Supports JSON formats for some objects as well. Resource loading, saving, and visual editing.
- `klamp't.vis`: Visualization.
- `Klamp't.control`: control modules.

You should also obtain the [Klamp't-examples Github project](#) for examples of how to run the Klamp't Python API. Example Klamp't data files are stored in the `Klamp't-examples/data`, and include

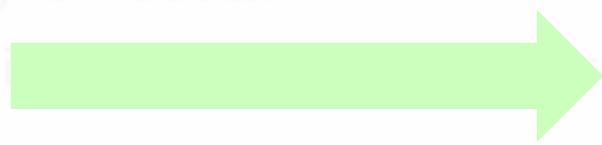
- `data`: world XML files
- `data/robots`: robot files
- `data/objects`: rigid object files
- `data/terrains`: terrain files
- `data/motions`: motions
- `data/resources`: an example resource collection
- `data/simulation_test_worlds`: worlds to test simulator functionality

# Kinematics

Open source – Klampt Python API

## IK Objective




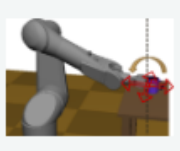
제한된 링크에 대한 제약 조건 정의

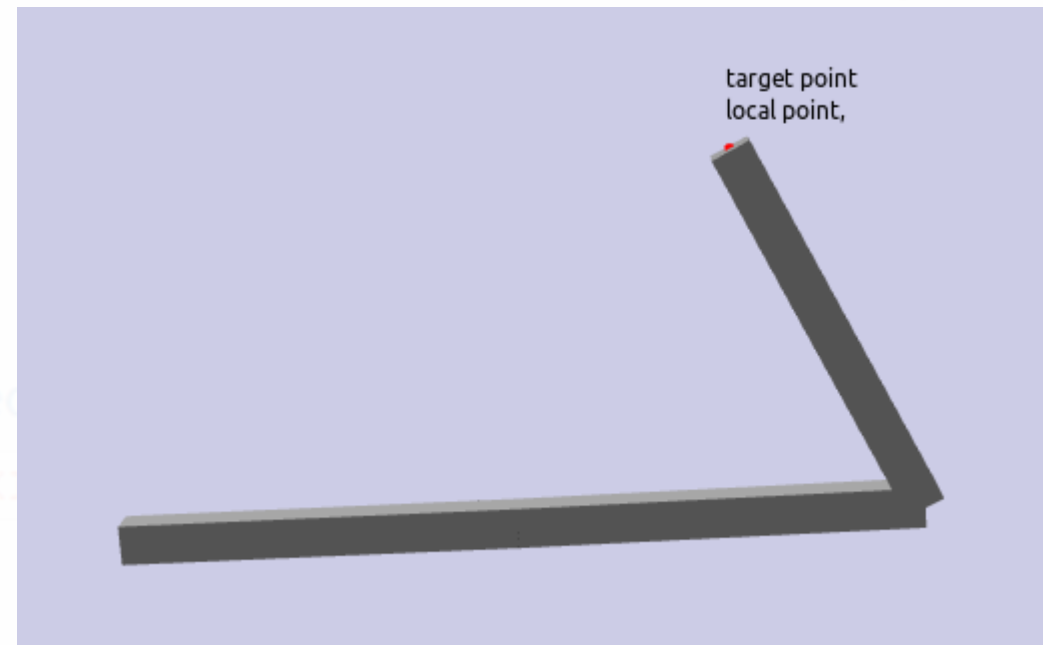


## IK Solver

목표에 지정된 제약 조건에 맞춰 계산

### Usual Objective Types

Constraint type	Point	Fixed	Hinge	Surface
Position	fixed	fixed	fixed	planar
Rotation	none	fixed	axis	axis
Constrained dims	3	6	5	3
Useful for	Point feet or fingers	Grasps, flat feet	Grasping cylindrical objects	Placing objects down with unspecified position and orientation
				



# 감사합니다!

---

쓰레기 자동 분류 솔루션 Invictus