

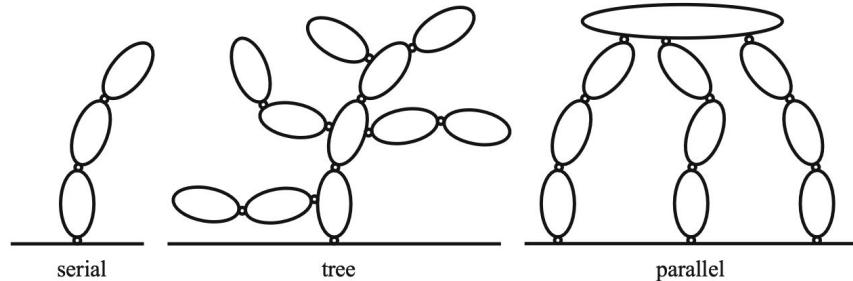


## Talent Mobility Program 2024

**Cable Driven Parallel Robots with  
unknown Anchor Points**

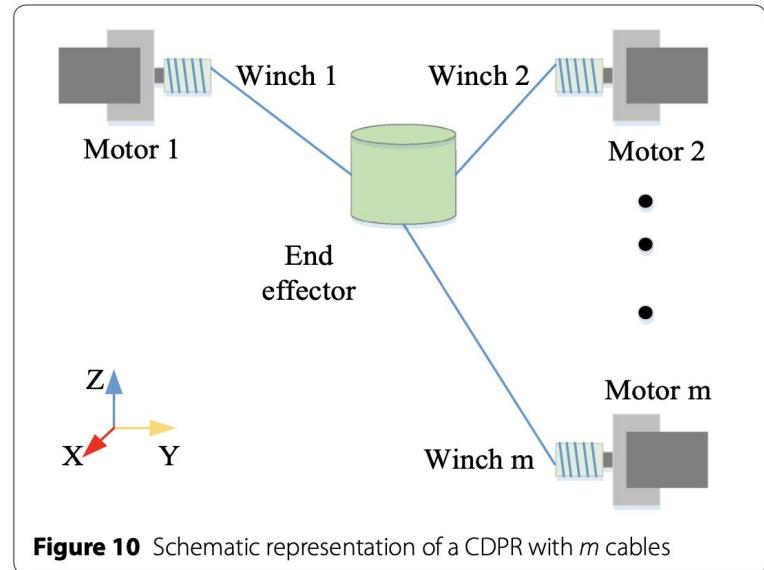
# Basics

## 1.1 From Serial Robots to Cable Robots



**Fig. 1.2** Comparison between the topological structure of serial, tree, and parallel robots

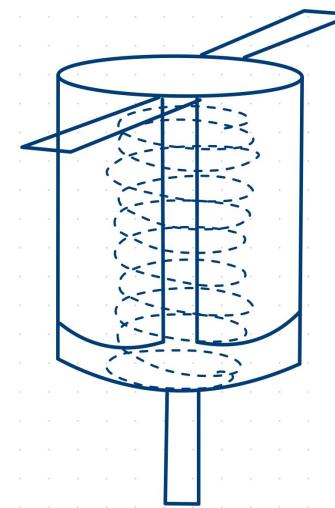
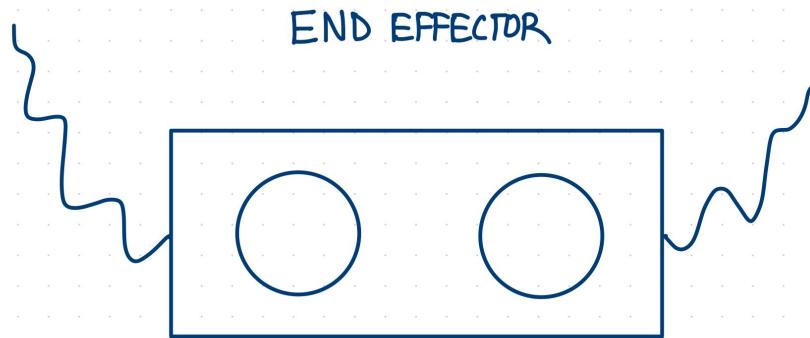
3



>>

# Scope

- End effector
- Projectile Launcher



>>

# End Effector

- Drawings
- Shaft Design : Stress concentration

## Mechanical Design

- ROS : Nodes used
- Python : as a programming language

## Software and Code

### Hardware used

- Dynamixel AX 12 motors
- Raspberry pi 4b
- INA260 power sensor
- TCA9548A I2C multiplexer

## Putting all together

### Theories

- Motor theory

>>

# End Effector

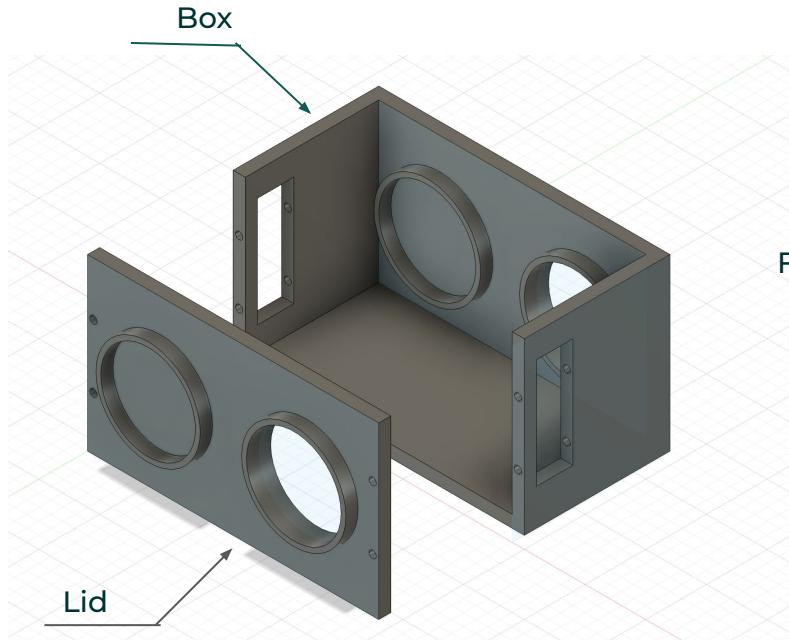
- Drawings
- Shaft Design : Stress concentration

## Mechanical Design

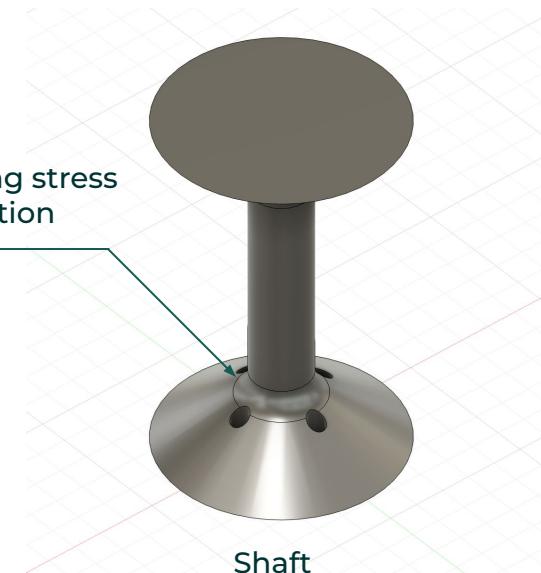


# End Effector

## Mechanical Design

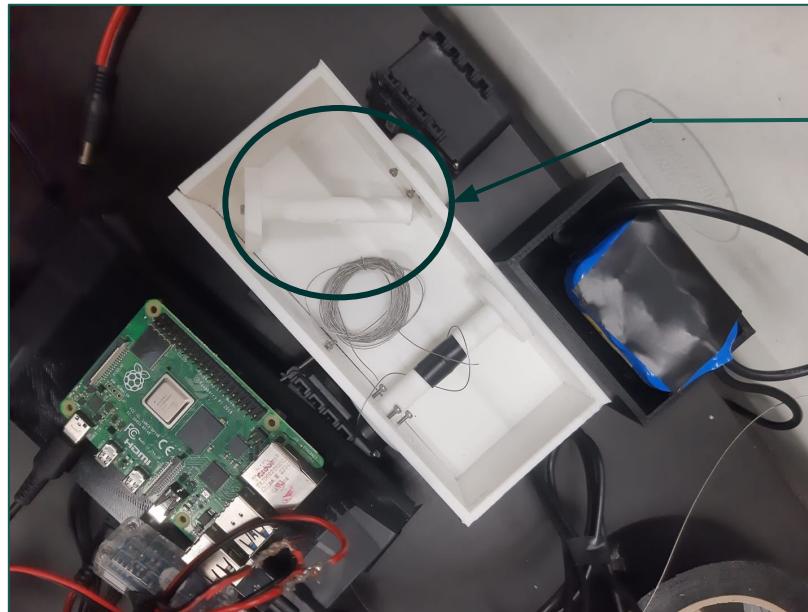


Fillet : Reducing stress concentration



# End Effector

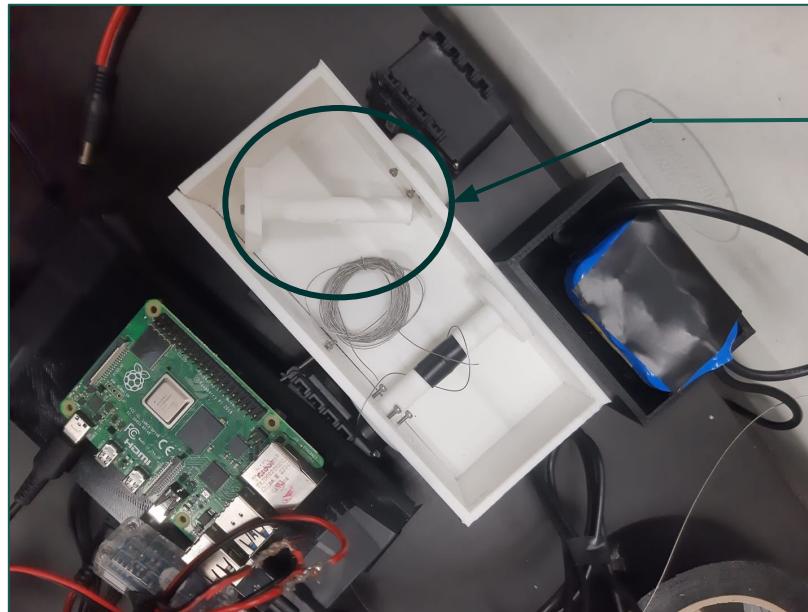
## Mechanical Design



Breakage due to Stress concentration

# End Effector

## Mechanical Design



Breakage due to Stress concentration

# End Effector

## Mechanical Design

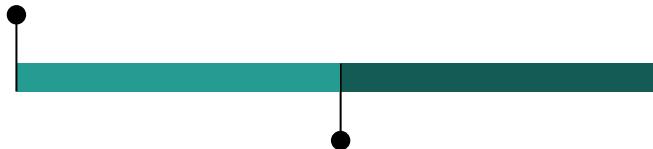


>>

# End Effector

- Drawings
- Shaft Design : Stress concentration

## Mechanical Design



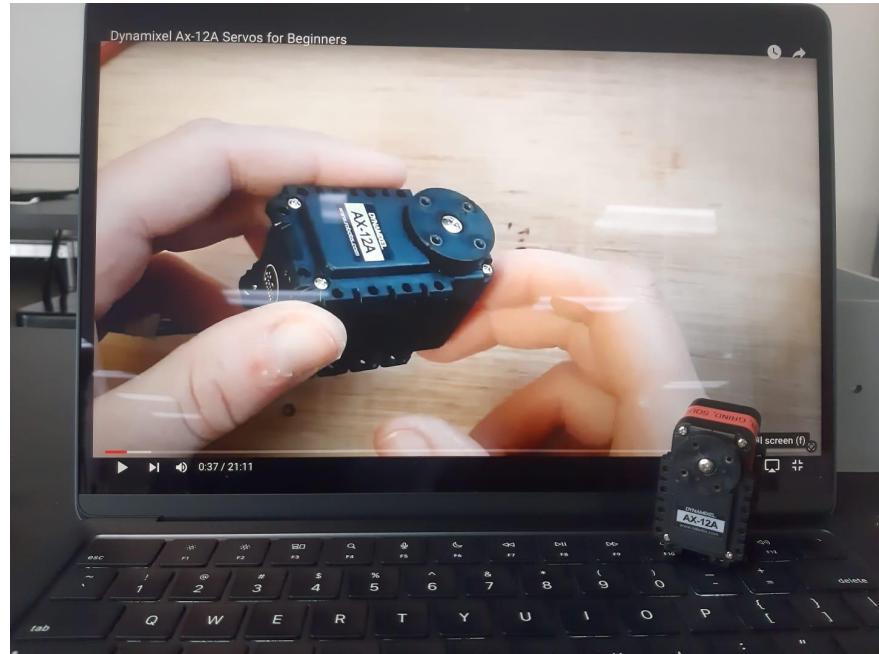
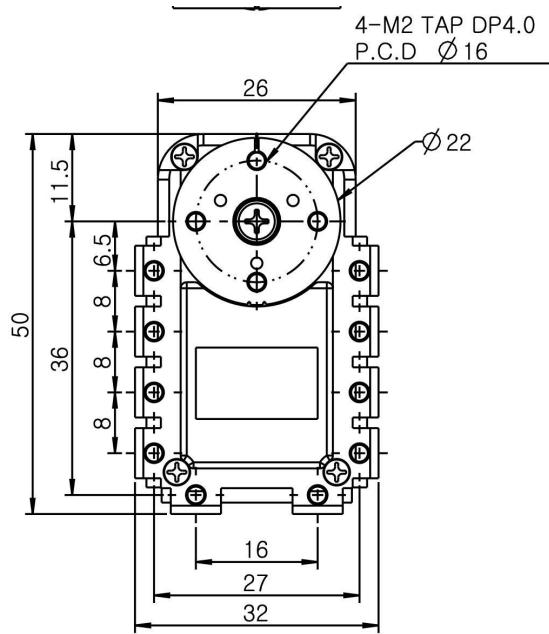
## Hardware used

- Dynamixel AX 12 motors
- Raspberry pi 4b
- INA260 power sensor
- TCA9548A I2C multiplexer

>>

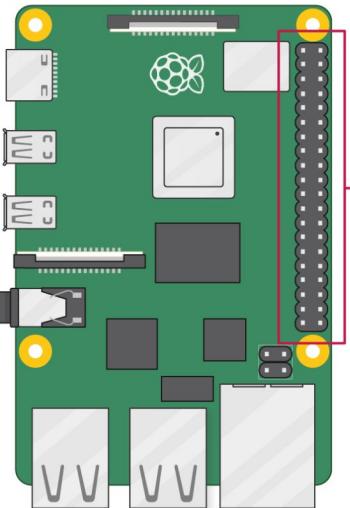
# End Effector

Hardware used :



# End Effector

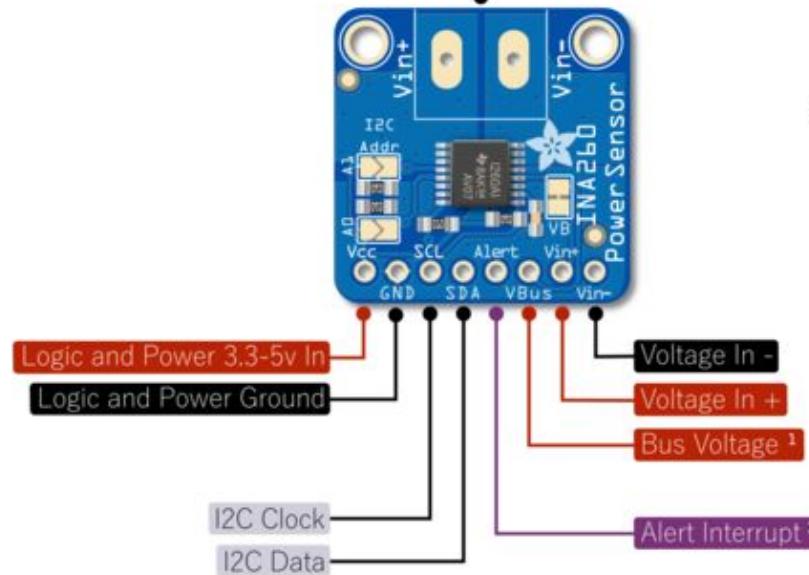
Hardware used : *Raspberry pi 4b*



3V3 power	1	2	5V power
GPIO 2 (SDA)	3	4	5V power
GPIO 3 (SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (TXD)
Ground	9	10	GPIO 15 (RXD)
GPIO 17	11	12	GPIO 18 (PCM_CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3V3 power	17	18	GPIO 24
GPIO 10 (MOSI)	19	20	Ground
GPIO 9 (MISO)	21	22	GPIO 25
GPIO 11 (SCLK)	23	24	GPIO 8 (CE0)
Ground	25	26	GPIO 7 (CE1)
GPIO 0 (ID_SD)	27	28	GPIO 1 (ID_SC)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM_FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM_DIN)
Ground	39	40	GPIO 21 (PCM_DOUT)

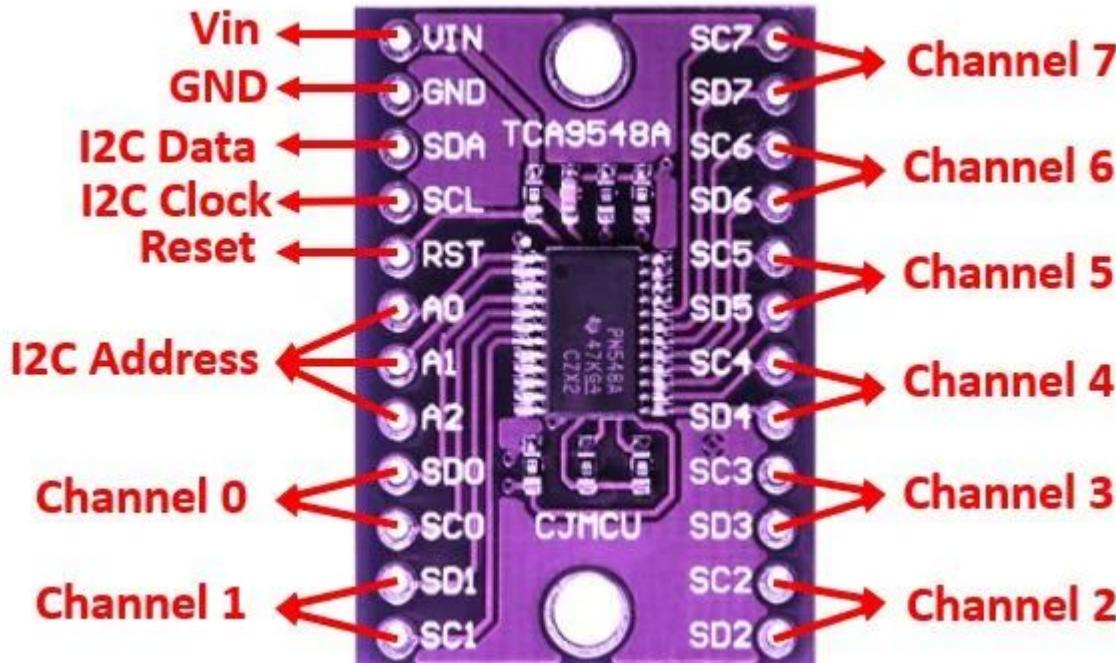
# End Effector

Hardware used : *INA260 power sensor*



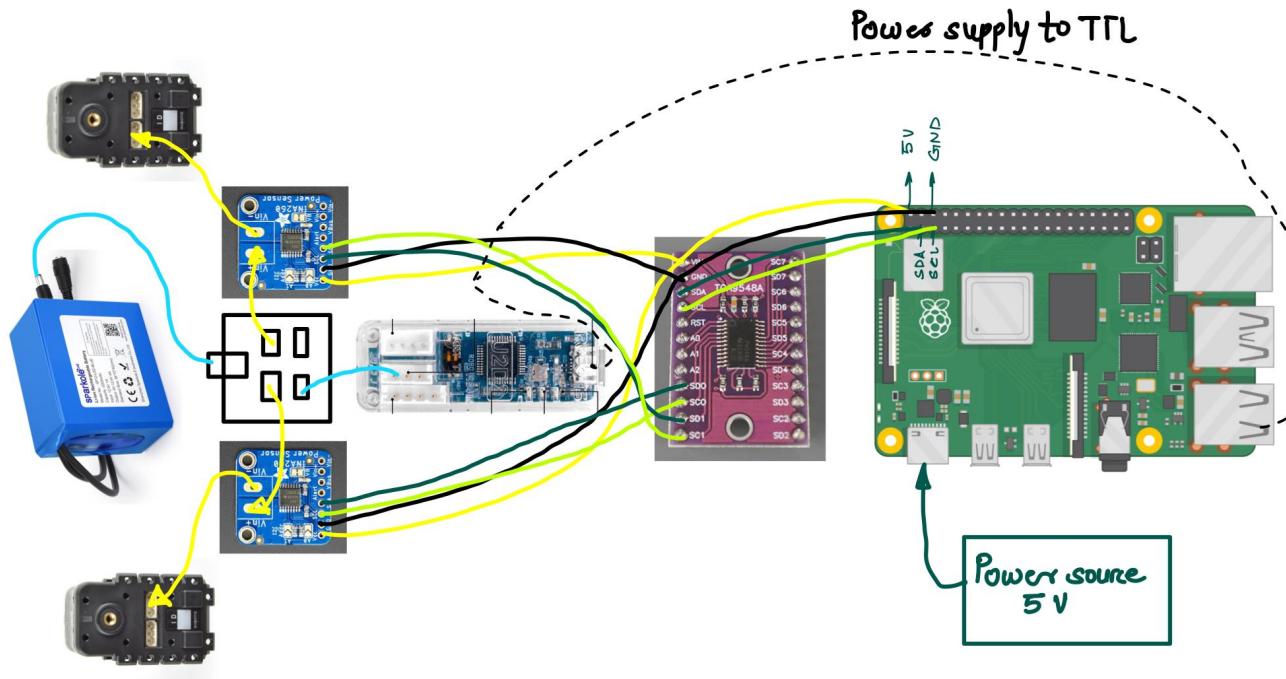
# End Effector

Hardware used : *multiplexer*



# End Effector

## Hardware used



# End Effector

- Drawings
- Shaft Design : Stress concentration

- ROS : Nodes used
- Python : as a programming language

## Mechanical Design



## Software and Code

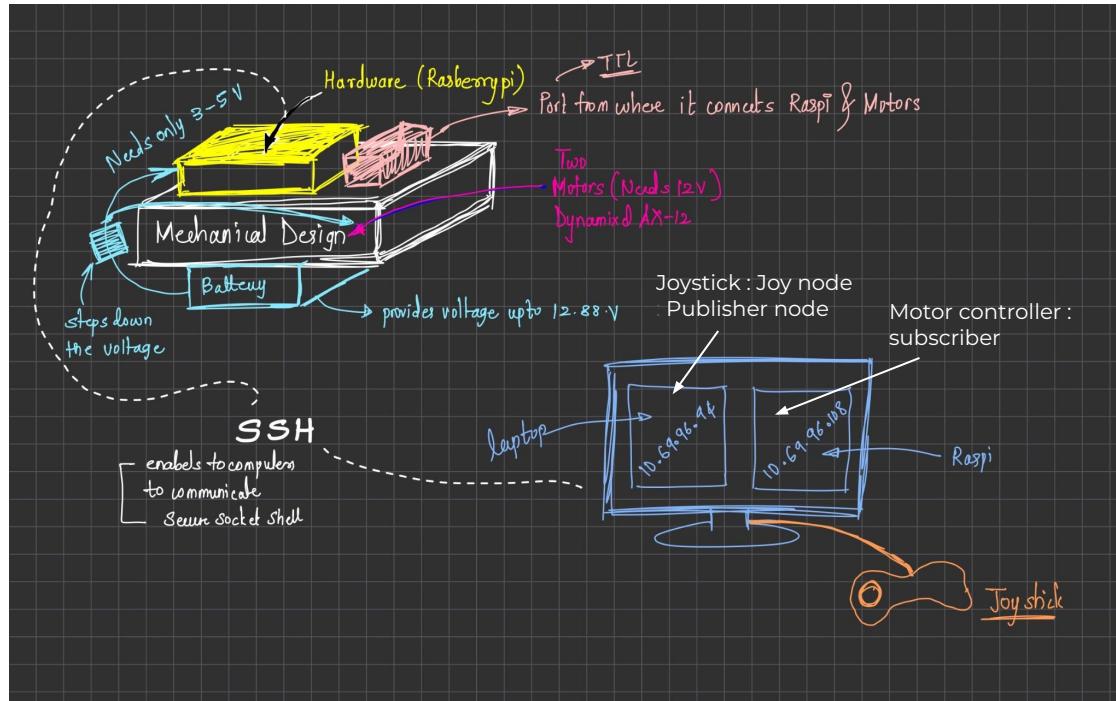
### Hardware used

- Dynamixel AX 12 motors
- Raspberry pi 4b
- INA260 power sensor
- TCA9548A I2C multiplexer

>>

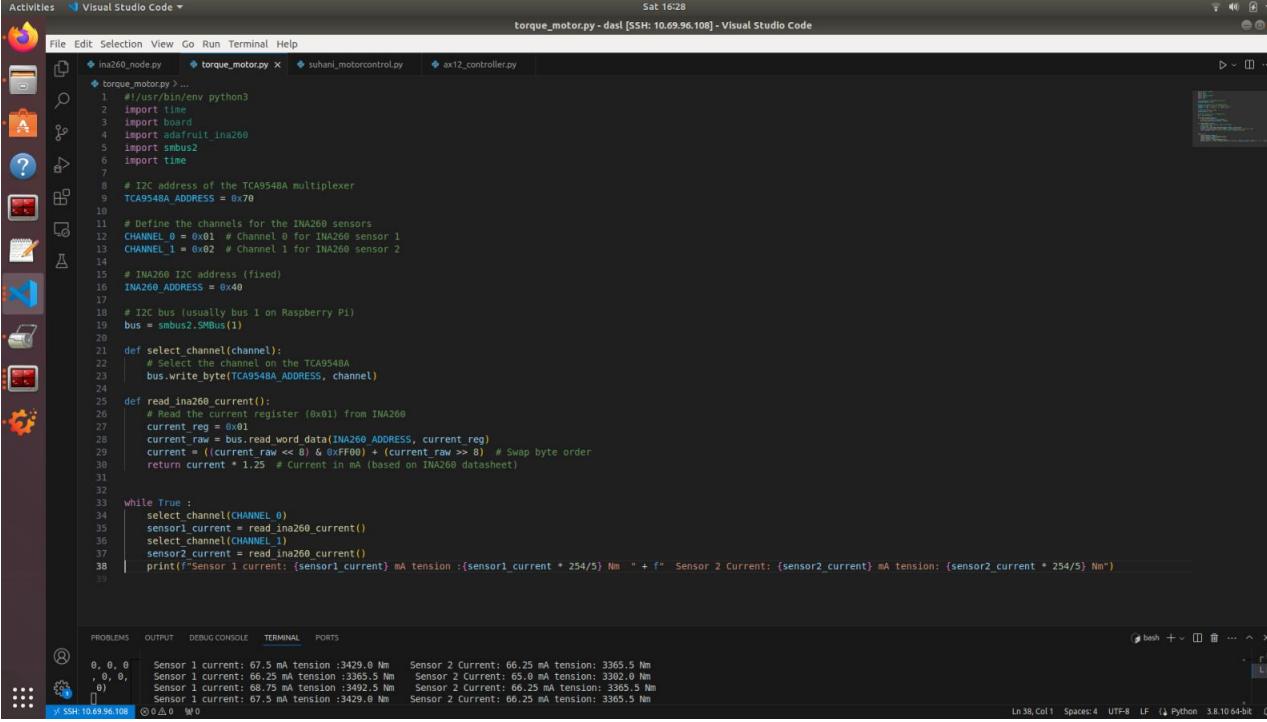
# End Effector

Software : code used : Communication



# End Effector

## Software : code used : Current sensors



The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Activities, Visual Studio Code, Sat 16:28, torque\_motor.py - dasl [SSH: 10.69.96.108] - Visual Studio Code
- File Explorer:** Shows files: ina260\_mode.py, torque\_motor.py (current file), suhani\_motorcontrol.py, ex12\_controller.py.
- Code Editor:** Displays Python code for reading current from INA260 sensors using I2C. The code includes imports for time, board, adafruit\_ina260, smbus2, and smbus. It defines I2C addresses for TCA9548A multiplexer and INA260 sensors, and implements functions for selecting channels and reading current values.
- Terminal:** Shows command-line output for four sensors (Sensor 1 and Sensor 2 for each channel). The output is as follows:

Sensor	Channel	Current (mA)	Tension (Nm)
Sensor 1	0	67.5	3429.0
	1	66.25	3365.5
Sensor 2	0	65.0	3302.0
	1	66.25	3365.5

**Bottom Status Bar:** Ln 38, Col 1, Spaces: 4, LF, Python 3.8.10 64-bit

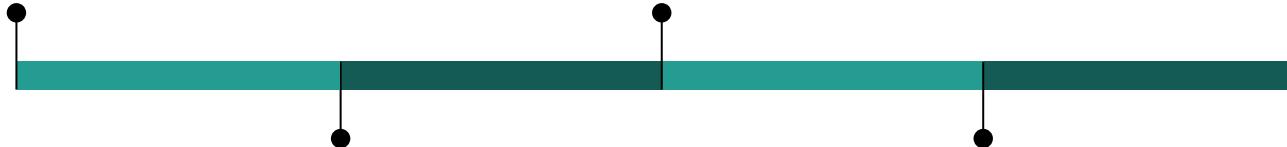


# End Effector

- Drawings
- Shaft Design : Stress concentration

- ROS : Nodes used
- Python : as a programming language

## Mechanical Design



## Software and Code

### Hardware used

- Dynamixel AX 12 motors
- Raspberry pi 4b
- INA260 power sensor
- TCA9548A I2C multiplexer

### Theories

- Motor theory

>>

# End Effector

## Motor theory

Current → Torque → Tension

- Specifications of motors →  
1. Stall Torque : 1.5 N.m (at 12V, 1.5A)  
2. No-load speed : 59 RPM.

3. Calculating the torque constant  $k_t = \frac{\text{stall torque}}{\text{stall current}}$

$$k_t = \frac{1.5 \text{ Nm}}{1.5 \text{ A}}$$
$$k_t = 1 \text{ N m/A}$$

4. Gear Ratio : 254 : 1 (What is gear ratio?)

5. Final equation :

$$\tau_{\text{output}} = k_t \cdot I \cdot (\text{gear ratio})$$
$$= \frac{1 \cdot I \cdot 254}{\boxed{\tau_{\text{output}} = 254 \cdot I}}$$

SI units.

Tension experienced by cables →

Cables are attached to shaft.

$$\text{Tension} = \frac{\text{Torque}}{5 \times 10^{-3}}$$

$$\tau = \bar{r} \times \bar{T} \therefore \text{Scalar values} \Rightarrow T = \frac{\tau}{r}$$

$\tau$  : Torque provided by the motor  
to rotate the shaft.

Radius of the  
shaft

# End Effector

- Drawings
- Shaft Design : Stress concentration

- ROS : Nodes used
- Python : as a programming language

## Mechanical Design



## Hardware used

- Dynamixel AX 12 motors
- Raspberry pi 4b
- INA260 power sensor
- TCA9548A I2C multiplexer

## Software and Code



## Theories

- Motor theory

## Putting all together



>>

# End Effector

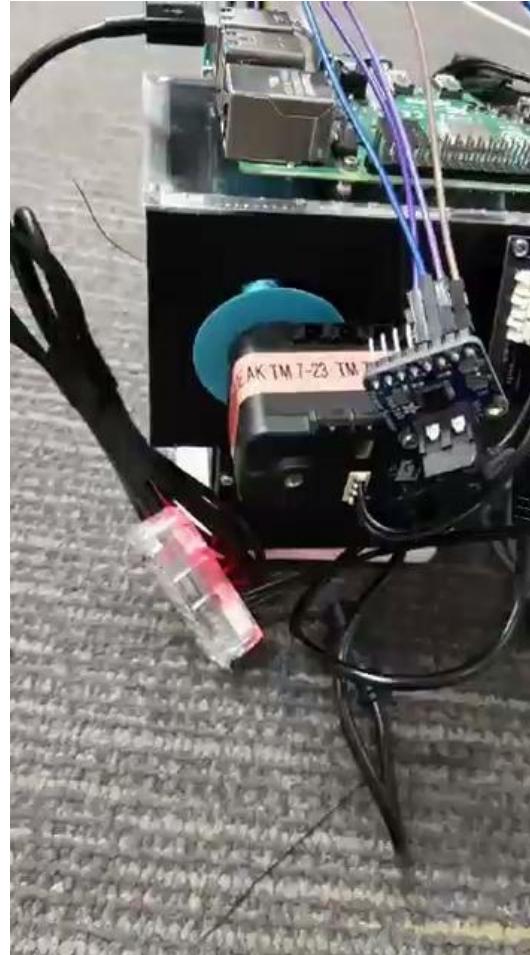
## Putting everything together



>>

## **End Effector**

Putting everything together



>>

# **End Effector**

Putting everything together



>>

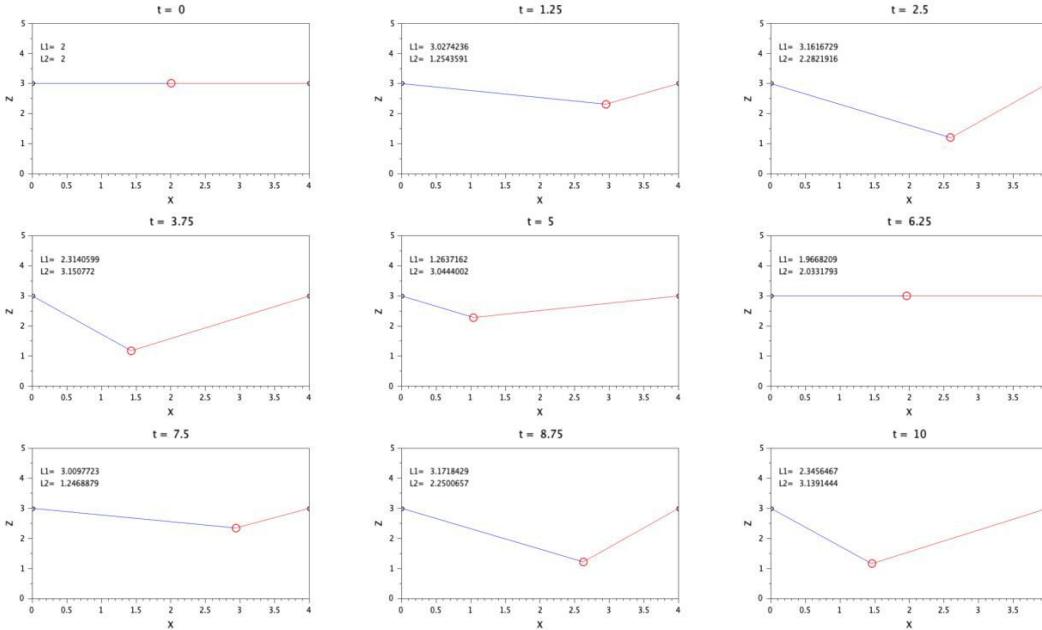
# **End Effector**

Putting everything together

>>

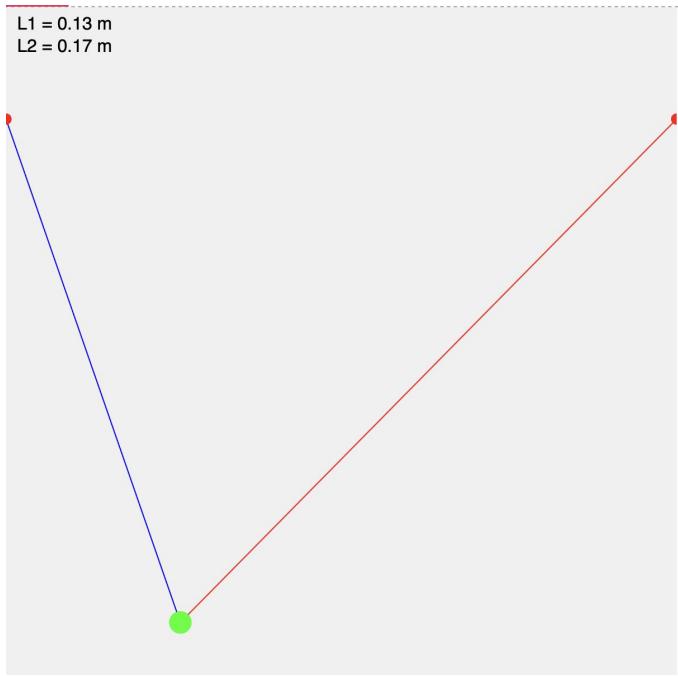
# End Effector

## Putting everything together



# End Effector

Putting everything together

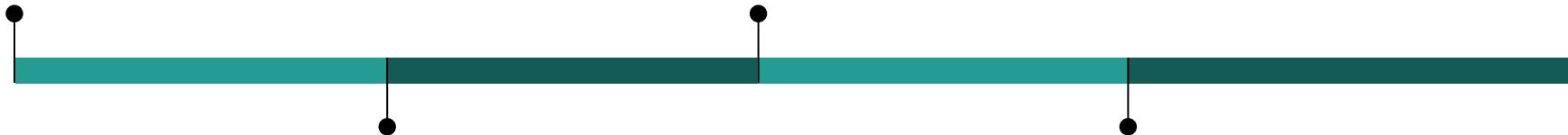


# Projectile launcher

- Spring constant calculation

Theory

Final Model



Mechanical Design

Anchors

>>

# Projectile launcher

- Spring constant calculation

## Theory



# Projectile launcher

## Working on spring constant

Given:  $h = 2\text{m}$ ,  $\frac{H-R}{2} = 1.35\text{m}$

To find: spring stiffness  $k$

Solution steps:

1. Solving projectile:
  - Horizontal motion:  $u \cos \theta = \text{horizontal velocity} = 2.11416 \text{ m/s}$
  - Vertical motion:  $t = 1.2771 \text{ s}$
  - Total initial velocity  $u = 6.611 \text{ m/s}$
  - Total final velocity  $v = \text{horizontal component} = u \cos \theta = 2.11416 \text{ m/s}$
  - using  $\theta = \text{vertical initial velocity} = 6.264 \text{ m/s}$
2. Solving energy eq<sup>n</sup>:  $\frac{1}{2} kx^2 = \frac{1}{2} mv^2 + mgh$  → Assumed values:
  - mass =  $50\text{g} = 0.05 \text{ kg}$
  - height =  $2\text{m}$
  - deformation  $x = 4\text{cm} = 0.04 \text{m}$
- After solving for  $k$ :  $k = 1366 \text{ N/m}$
3. Solving / Identification of  $k$  for the existing spring →

From spring design: 
$$k = \frac{G d t}{8 D^3 n}$$

After solving:  $k = 11200 \text{ N/m}$

specifications for existing spring:

Data: → (Measurements are taken manually)

1. Outer diameter ( $D_{\text{outer}}$ ):  $8.80 \text{ mm}$
2. Inner diameter ( $D_{\text{inner}}$ ):  $7.20 \text{ mm}$
3. Number of coils ( $n$ ):  $11$
4. Free length:  $35 \text{ mm}$  (Not directly needed for calc.)
5. Material Shear Modulus ( $G$ ):  
Material is Zinc plated steel.  $G \approx 77 \times 10^9 \text{ Pa}$

**Conclusion :**

Theoretical value of  $k$  considering the projectile & spring energy & cubical dimensions →

$$k = 1366 \text{ N/m}$$

Value of  $k$  determined using spring dimensions (using the existing spring) is

$$k = 11200 \text{ N/m}$$

Springs in series →

→ 

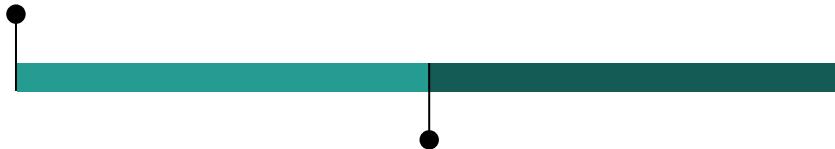
here  $k_1 = k_2 = k$

$$\therefore \frac{1}{k_{\text{total}}} = \frac{1}{k} + \frac{1}{k} = \frac{2}{k}$$
$$\therefore \boxed{k_{\text{total}} = \frac{k}{2}}$$
$$k_{\text{total}} = \frac{11200}{2} = 5600 \text{ N/m}$$

# Projectile launcher

- Spring constant calculation

## Theory

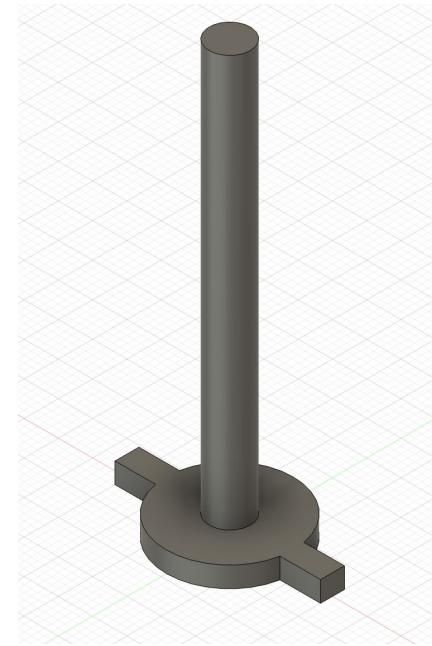
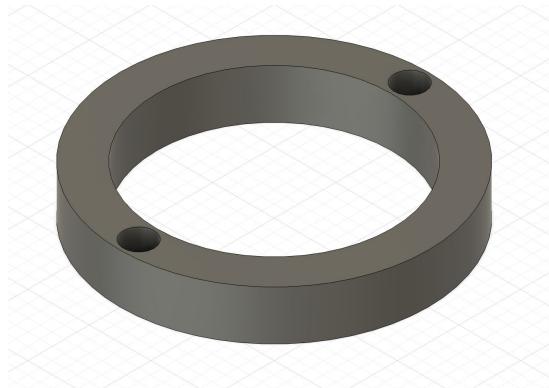


## Mechanical Design

>>

# Projectile launcher

## Design



>>

# Projectile launcher

- Spring constant calculation

Theory

Final Model



Mechanical Design

>>

# Projectile launcher

Final model



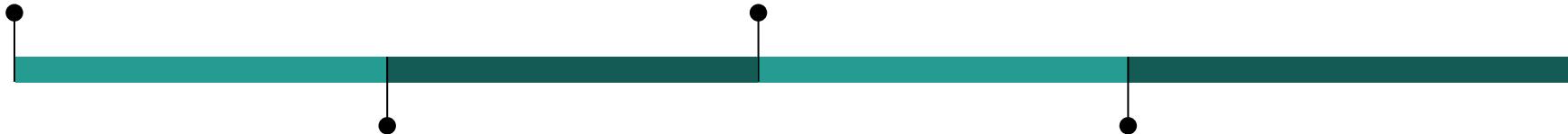
>>

# Projectile launcher

- Spring constant calculation

Theory

Final Model



Mechanical Design

Anchors

>>

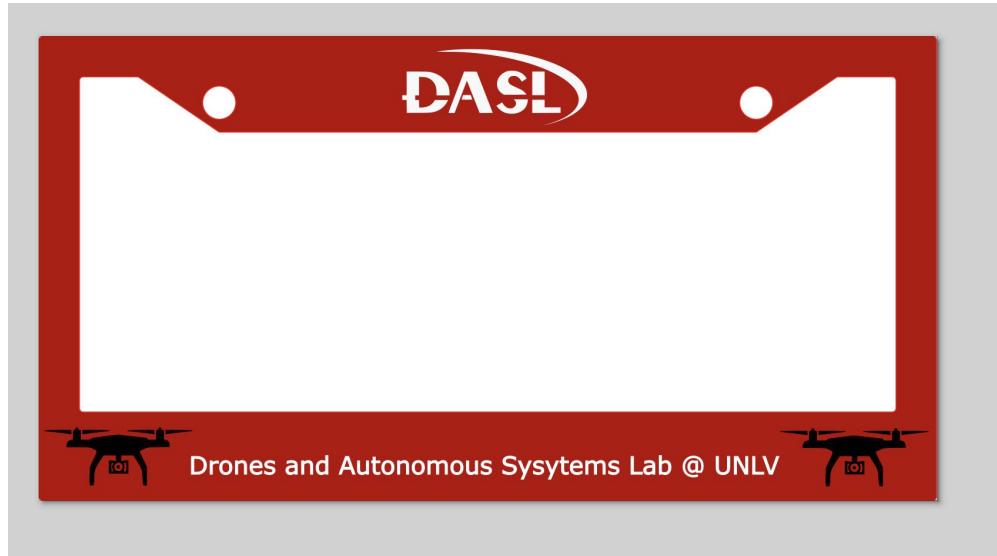
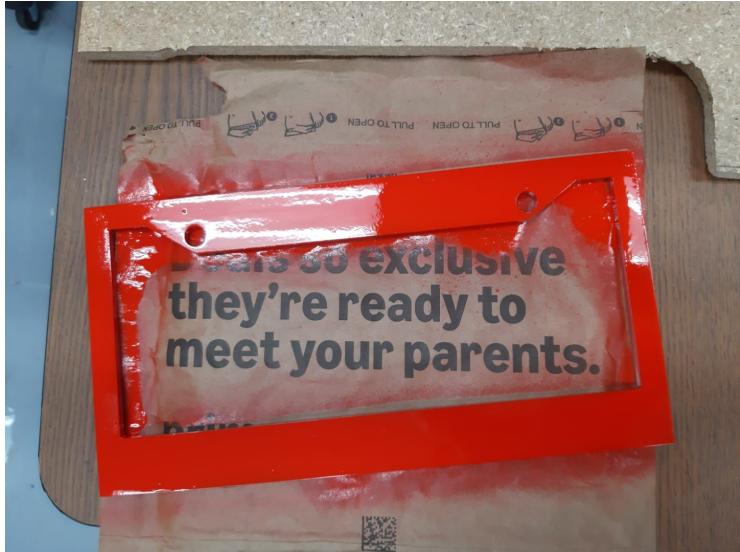
# Projectile launcher

## Anchors



>>

# Unwinding



>>

# Unwinding



>>

# **Thank You**