Modular Tendon-Driven Soft Robot

Design & Assembly Guide

Project Complexe 3 – Group 20

IMT Atlantique

March 24, 2025

Contributors:

- Emna Adhar
- Alexian Bothorel
- Yuri Naves de Oliveira Araújo
- Achylle Pascanet

1. Project Background and Objectives

1.1 Background

The project meets the LS2N laboratory's need to design a soft robot powered by tendons. This robot, inspired by natural structures such as elephant trunks and octopus tentacles, has a flexible structure controlled by a network of tendons. It is particularly well suited to applications requiring access to confined spaces and delicate handling. These applications include minimally invasive surgery in the medical field, industry, underwater robotics and agricultural harvesting.

1.2. Operating principle

The robot is made up of 8 identical modules:

- 4 translation modules for pulling on wires
- 4 torsion modules to turn the wires
- Each wire is controlled by a combination of translation and torsion, providing all the degrees of freedom needed to manipulate the soft robot.

There are also 4 rails and common components to bring the modules together

1.3 Resources required

- 8 three-phase motors (to be characterised: torque, rated speed, dimensions, supply voltage, etc.)

- 1 AC power supply (specifications to be specified according to motors and ESCs)

- 8 bi-directional ESCs (Electronic Speed Controllers)

- 2 Raspberry Pi Pico boards

- Plastic wires (type to be specified)

- 1 3D printer

- 4 fishing lines (0.3 mm nylon)

- Screws (one M6 X 14 SOCKET HEAD box and one M3X10 box) and corresponding nuts

2. Engine control

2.1 Electronic part

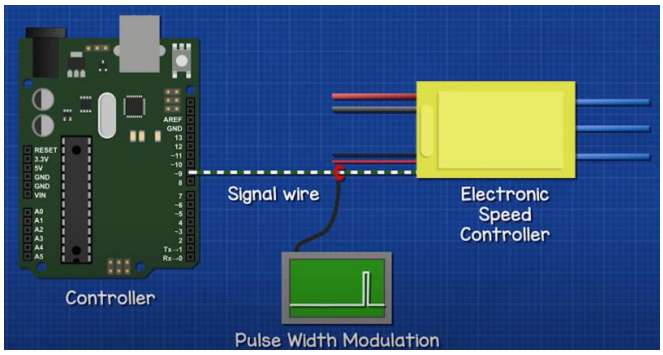
The motors used in our robot are three-phase brushless motors. They are each controlled using an ESC (Electronic Speed Controller), the purpose of which is to take a servomotor PWM as input and distribute 3 phase-shifted signals, corresponding to the command, to the 3 motor inputs. These are of the following form:

These have :

- 2 power cables (red and black) to be connected to the GBF or to the planned power source.
- 3 cables (black) to be connected to the three motor inputs (the order in which the cables are connected affects the direction of rotation of the motor).
- 3 cables for controlling the motor, with a black GND cable, a red power cable (only used when the motor operates with a potentiometer) and a cable (white or other) for controlling the speed and direction of rotation of the motor using a servomotor-type PWM.

The maximum amperage supported by our ESCs is 1.2A, which in practice was never approached during the project. What's more, the voltage we used at the ESC input was around 20V.

2.2 Control signal section



The ESC therefore requires a PWM (Pulse Width Modulation) with a frequency of 50 Hz and a high period of between 1 and 2 ms. 1.5 ms being the neutral position (the motor does not move) and with the motor rotating in one direction between 1 and 1.5 ms, then in the other

direction between 1.5 and 2 ms (this direction of rotation can simply be changed by swapping two motor supply cables).

In our case, we don't need a high rotation speed and we have therefore stayed within the bounds of [1.4;1.6]ms for the high part of our PWM. However, as our modules have variable performance for longitudinal displacement, due to friction that can vary from one module to another and directly to changing motor performance, we have had to use different high periods and can therefore not generalise the commands to each motor. This problem should disappear with the integration of regulation into the system.

2.3 Code

All you need is a PWM signal to control a motor, so you can easily control all 8 motors with a single microcontroller (in our case, a Raspberry Pico W). The basic micro-Python code used to generate this signal is as follows:

```python
from machine import Pin, PWM

# Configuration de la broche PWM
pwm_pin = Pin(15)  # Remplacez 15 par le numéro de votre broche
pwm = PWM(pwm_pin)
pwm.freq(50)  # Fréquence de 50 Hz (période de 20 ms)
pulse = 1500


def set_pwm_pulse_width_us(pulse_width_us):
    """
    Définit la durée du signal haut en microsecondes (entre 1000 et 2000 us).
    """
    # Calcul du duty cycle en fonction de la période de 20 ms (20000 us)
    duty_cycle = int(pulse_width_us / 20000 * 65535)
    pwm.duty_u16(duty_cycle)

set_pwm_pulse_width_us(pulse)
```

This operating system can simply be multiplied to run several motors in parallel (see Scenario4Motors.py and Scenario4MotorBis.py).

2.4 Other systems explored

During our project we also tried to use other means to control our motors, which didn't work for different reasons. These were :

- Launchpad and Booster by Texas Instrument
- Escon Module by Maxon Motor
- AEDM encoder by Avago Technologies

Launchpad and Booster by Texas Instrument

The original project from which the project's mechanical model is taken (see appendices) uses Texas Instrument Launchpads and Boosters to operate the motors. However, we didn't manage to get them to work to our satisfaction because the documentation was too sparse and difficult to understand. We therefore abandoned this solution.

Escon Module by Maxon Motor

We then tried using an Escon module from Maxon Motor, which we didn't keep because the interface proposed by the brand made using the motors not very intuitive and parallelizing the motors complicated. We then switched to the Raspberry Pico, bearing in mind that it's also possible to swap it with an Arduino board as they have very similar operating modes.

AEDM encoder by Avago Technologies

Finally, we also tried to implement the AEDM encoders present in the original project to be able to carry out closed-loop regulation to control our motors. We therefore developed a code to count the pulses detected by the encoder and to determine the direction of rotation (see Encoder.py), but we were unable to implement it because of excessive errors in the measurements taken. These are probably due to a problem of slippage between the motor and the shaft, and then between the shaft and the encoder.

3. Development of the mechanical part

3.1. Development and production of parts

Most of the components were already pre-designed and our team simply had to adapt some of them or create new ones to increase the robot's robustness and resistance.

All the technical drawings are in our drive, with not only the drawing of each component, but also the representation of its assembly, made in Inventor AutoDesk. To access the drive, simply . The same link leads to the assembly manuals for the actuator modules and the sensor assembly.

To reproduce a project like this, you first need to print the structural elements using a 3D printer. You can view and find out about all the components required by consulting the "Complete assembly" file in the "All elements" folder.

3.2. Assembly procedure

Next, it is necessary to observe the order of assembly of the actuator modules, as indicated in the manual.

Once the actuator modules have been correctly assembled, the following order of assembly is recommended for the robot:

1. Assemble "Wall_of_platform" with "Head_of_platform".
2. Attach the rack support ("Support rack") to the front base ("Head_of_platform").
3. Insert the pillow blocks into the rails.
4. Insert the rail supports into the front base and screw them in place.
5. Insert the carriages already assembled with the modules into the rails.
6. Insert the rack supports in the rear base ("New short base") (this is an improved version that we decided to use in our project)
7. Connecting the rear base to the front base + rail assembly

8. Once the two halves are assembled, place them one on top of the other and fix them in place with bolts and nuts.
9. For the arm, use a 3mm carbon fibre beam and glue the discs together with strong glue according to the distance between the discs that you think is ideal. We used 12 cm between each disc. Next, attach the fishing line to the last disc and secure it to the front actuators. To do this, we drilled a hole in the 'shaft_gear' that allows the fishing line to be attached and detached.

To attach the arm to the robot, we drilled 2 diametrically opposed holes in the base disc, and 1 hole in each front base to attach the base disc to the structure using a screw and nut.

4. Procedure

Procedure for putting our robot into operation :

1. Connect the motor to the two-way ESC.
2. Connect the ESC to the power supply and to the appropriate pins on the Raspberry Pi Pico board.
3. do the same for each engine
4. In the Thonny environment, write the code in MicroPython (see 2.3).
5. The code is used to change the speed and direction of rotation of the motor.
6. Download the code onto the Pico card.
7. Mechanical assembly of parts (see 3.2).
8. Add the 4 trolleys to their respective rails, paying attention to their orientation.
9. Closing the case and adding the arm.
10. Placement and tensioning of the tendons at the desired vertebrae.

5. Conclusion

In conclusion, this project provides a solid basis for the design of a flexible, tendon-driven robot that is reproducible and viable, and whose structure is highly customisable thanks to the 3D printing of its components. This type of system offers many potential applications in a wide range of fields, including medical and industrial applications. In addition, the system could be upgraded by replacing the wires with rigid rods, driven not only in translation by the existing motors, but also in torsion by the addition of dedicated motors. This configuration would allow the robot to be rotated around itself, considerably extending its range of movement. However, to guarantee its efficiency and reliability, further studies are still required, particularly with regard to the implementation of regulation and the development of a suitable control model.

Appendices

Documentation on a project using the same motors: https://github.com/open-dynamic-robot-initiative/open_robot_actuator_hardware/tree/master

Links to the code documentation for the same project :

https://open-dynamic-robot-initiative.github.io/documentation_portal/

Documentation of another project similar to ours:

https://github.com/ContinuumRoboticsLab/OpenCR-Hardware/blob/main/mechanics/LOTR_TDCR-spatial/README.md

Encoder datasheet: https://www.mouser.com/datasheet/2/678/avgo_s_a0001422768_1-2290945.pdf?srsltid=AfmBOoo6IEAnr1tjLinl_LVz-lsKy8QvK8D0zakHaQNjzNFqGXHODZ3Z

Datasheet for the pico card:

https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf

File with mechanical drawings: https://drive.google.com/drive/folders/1-UzpPLCmslqcSs-0DEPWHgyKrePXdWNa?usp=drive_link

Assembly manuals: Modular actuation unit https://drive.google.com/file/d/1cwoCN-Bz9DuOh9sWZBFv4c0yCU_1f9Yh/view?usp=drive_link

Encoder assembly https://drive.google.com/file/d/1M4ncAQqrbUTjT0UvDa6qT_8cq7Nisqg9/view?usp=drive_link