Electronic Control System for a Tendon-Driven Continuum Robot

Charlene Chen, Lara Yanc (Group 19)

ABSTRACT

COMP0207 Introduction to Electronics Project Report outlining a extrinsically-actuated tendon-driven continuum robot controlled by a set of four buttons. The report outlines circuit components, control system flowchart and pseudocode, and electric circuit design.

Keywords: Control Circuit, Extrinsic Actuation, Continuum Robot, Electrical Design

1 INTRODUCTION

Differing from traditional notions of robots modeling human behavior such as humanoids and industrial robots, continuum robots present an alternate architecture which emphasizes ranges of motion unachievable by humans and vertebrates due to the limitations introduced by skeletal structures, opening a new realm of motion resembled by biological structures of invertebrates – elephant trunks, tentacles, et cetera [4]. Continuum robots manifest the unique characteristics of muscular hydrostats in several common forms: cable-driven, concentric-tube, and soft continuum robots [4]. Coupled with sensing, control, and actuation systems that support the main operations of the device, continuum robots can be used for inspections and operations in specific environments less navigable by humans, such as the human body, zero-gravity, underwater, or generally confined, hazardous spaces. This allows for applications in medical fields with minimally invasive surgeries and space exploration.

Two major variations of sensing systems derived from the targets desired by control systems within continuum robots are the following: environment perception and shape perception. According to Sincak, environment perception focuses on the interactions between the robot and the environment using sensors embedded on the outer layers of the robot, while shape perception aims to achieve proprioception of the robot body and reaching a specific configuration [5]. The principles which determine the robot's configuration are the degree of rotation and extension or compression of each vertebrae. By manipulating the forces pulling on each tendon of a cable-driven continuum robot, combinations of strings being pulled will reorient the robot body in response to the environmental feedback, whether the feedback is distance from the target or pressure signaling the device to avoid an obstacle in its path.

To achieve the first stage of continuum robot control, there are a few methods for optimizing how accurately the robot reaches its desired configuration upon receiving a command. The general approach of translating motion modeled in simulation state spaces to real-world action spaces utilizes reinforcement learning on Proportional-Integral-Derivative (PID) controllers [6]. State spaces can contain infinite configurations reflecting the continuous nature of the robot [3], thus in order to narrow the degrees of freedom to ones applicable to the task, a set of chosen parameters used to control the robot are selectively optimized [1]. For our project, the parameters to be optimized are relative to movement along the horizontal (x), vertical (y), and yaw (z) axes.

In this report, we will be outlining the fundamental tasks to achieve the first stage of control. Our control system will feature microcontroller units (MCU) and motor-drivers (both of the STM32 Nucleo class) attached to four motors, where the encoders on each motor allows us to make the robot reach a specific configuration, such as bending 90 degrees along the x-z plane [2]. A four-button control will be used to manually control the robot's bending angle, rotation, and extension/compression, alongside LEDs located at the robot base serving as a visual feedback to the user on the degree of each motion characteristic mentioned.

2 THEORETICAL BACKGROUND

As explained in the introduction, the system utilizes four motors, each of which connecting to a string that can be wound around a pulley each to extend and contract the system, allowing it to move with up to 6 degrees of freedom. The system operates mainly by button control located on the uppermost layer of the actuation unit, each controlling one direction. While moving with respect to these two operations, the robot also displays the status of each motor using an LED for each motor which is brighter when contracted and dimmer when the corresponding motor is extended. Algorithm 1 shows the pseudocode for the robot operation.

Algorithm 1 Pseudocode for Button Control

```
while $system$ is activated do

    if $input == LEFT$ then
        // Wind strings controlling -x axis to exert force;
    else if $input == RIGHT$ then
        // Wind strings controlling +x axis to exert force;
    else if $input == DOWN$ then
        // Wind strings controlling -y axis to exert force;
    else if $input == UP$ then
        // Wind strings controlling +y axis to exert force;
    end if

    // Update LED outputs based on a modified odometry

end while
```

The pseudocode shown in Algorithm 1 assumes the system equation is essentially unknown. From a programming and control aspect, this is the safer approach as the modular nature of the continuum robot makes it difficult to predict how the final robot will behave according to varying inputs. However, it is worth to note that the system follows Hooke's Law in terms of being an oscillating system affected by an outside force. Thus, we know that the system will follow the overall equation $F = -kx$ where F is the resistance force generated by the springs, k is the combined spring constant of all springs and x is the displacement of the tip of the robot (the center of the final node) with respect to the neutral position. As mentioned, while the system equations are known, the particular parameters are difficult to ascertain and the algorithm provided (in conjunction with reliable encoder inputs) will allow for control of the system as an unknown system while ensuring stability and protecting the degrees of freedom.

The system utilizes two microcontrollers (to be specified in the Electronic Composition Section) to control a pair of motors using the specified motor-driver. The output of the motors will be recorded and used as the input to control the status LEDs.

By using a set of four buttons corresponding to each tendon and direction ($\pm x$ and $\pm y$), the signal can be kept to digital, ensuring the digital ports in the microcontroller and motor driver can be used to control the motors without requiring additional communications between the microcontrollers or constant computer/manual input.

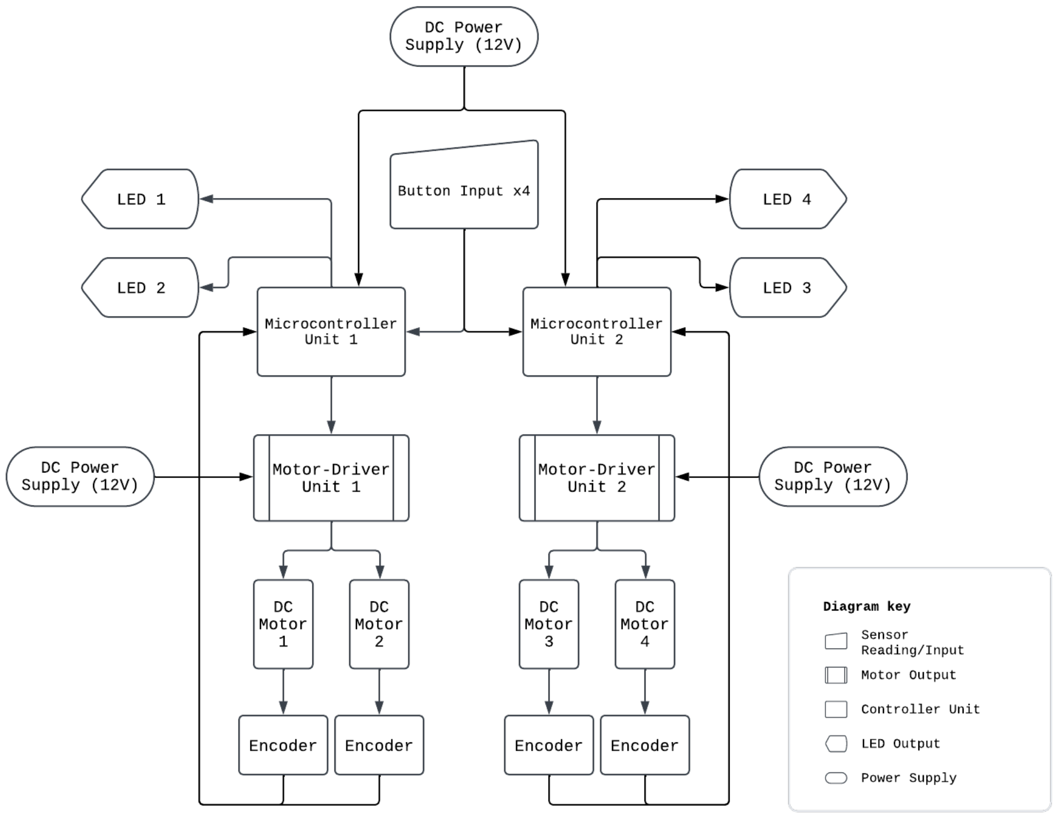


Figure 1. Diagram depicting the circuit composition.

3 ELECTRONIC COMPOSITION

Component	Amount	Explanation
DC Power Supply (12V)	3	One power supply per pair of motors One for both microcontrollers
LED	4	Outputs indicating the state of the corresponding motor depending on encoder input
Button	4	Four buttons capable of receiving input for two axes and activate preset functions
Microcontroller	2	NUCLEO-F303ZE to be connected to X-NUCLEO IHM12A1 Motor Driver per pair of motors
Motor Driver	2	One X-NUCLEO IHM12A1 to control per pair of motors
DC Motor	4	DIGILENT 290-006 DC Motor with 1:19 Gearbox with embedded 6-cable output encoders
Breadboard	1	Standard Breadboard

Table 1. Electronic Components

Table 1 outlines the expected amount of components to be used during construction. The datasheets and pinout diagrams of the components ensure that the circuit's usage will be within parameters and all relative integrated circuit components are compatible with one another.

4 ELECTRICAL AND ELECTRONICS DESIGN

As can be seen in Figure 2, the circuit consists of two sets of the following composition: one microcontroller and motor-driver unit, two motors with embedded encoders, two power supplies, two input points, one potentiometer and two LEDs for each pair of motors.

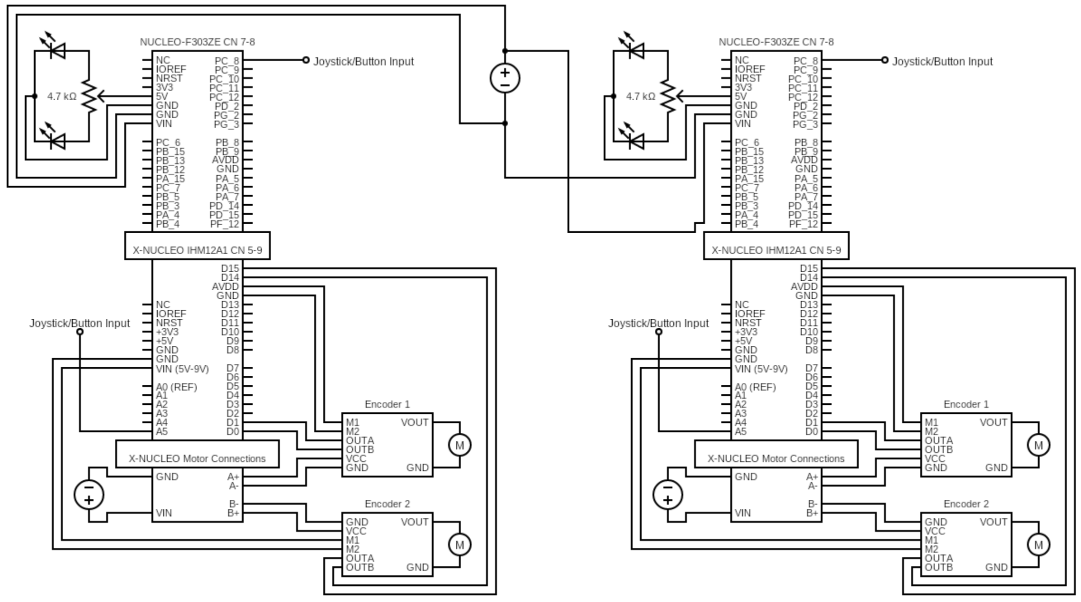


Figure 2. Full MCU Circuit Diagram

Figure 2 provides the circuit diagram of the full system consisting of two microcontroller, two motor-driver and four motors, which will be constructed to create the flow shown in Figure 1. It is of note that due to software limitations, the integrated circuit components shown in Figure 2 only show the available pinout of the combined microcontroller-motor driver unit and depending on the particular model of the units utilized, the relative locations of the ports used may change. Similarly, depending on construction decisions and limitations, the particular nature of the button or joystick may vary; however, the input provided will be a digital signal to ensure compatibility with the PWM ports of the microcontroller.

This design provides a simple solution to allow for control over the 6 degrees of freedom attainable by the robot. By separating the horizontal and vertical axes across the choice of four button inputs and relaying the same input to the two distinct microcontrollers, the system is able to avoid adding a communication system between the microcontrollers while maintaining the ability to synchronize motor movement. Overall, the system utilizes the encoder and microcontroller capabilities to allow the continuum robot to move with 6 degrees of freedom and return to home position from any state.

5 EXPERIMENTAL EVALUATION

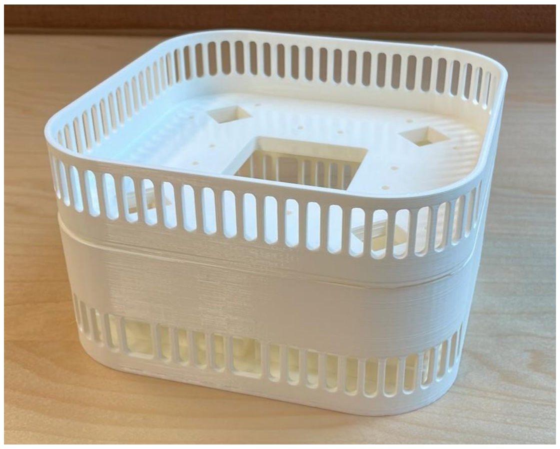


Figure 3. Image of the Empty Base.

Figure 3 shows the constructed base before the addition of the electronic and other physical components. As can be seen, the base was constructed to fit all three stages well and be large (and heavy) enough to stay stable despite the weight of the continuum robot moving. Several design choices were made for the base that allowed for successful construction: the major examples being the slits in the outer walls that allow for cables and zip ties to pass through, appropriately designed gaps that allow us to stabilize the batteries and motors as necessary, and a transparent top layer that allows visibility through to the motors and the second stage.

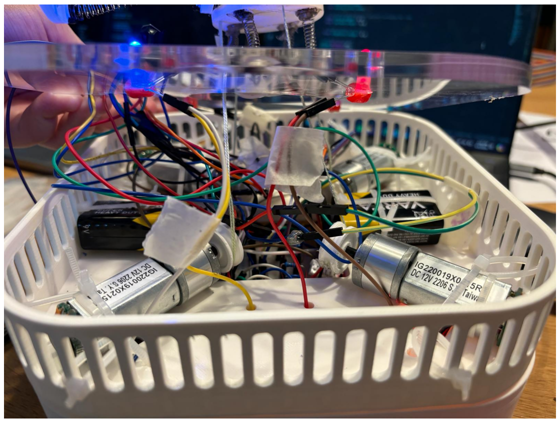
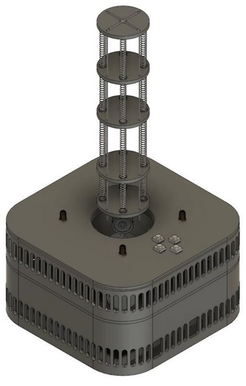


Figure 4. Image of the Second Stage of the Base.

Figure 4 shows the middle stage of the base, with the microcontrollers (with motor shield overlays) below and connected to the batteries and DC motors shown in this stage. The lower stage also contains a breadboard which is used to connect the buttons that act as a joystick.

Figures 5a and 5b show the final assembly in design and in real life, respectively. As can be seen, the physical assembly matches the design closely with a few changes mentioned above. Due to time and material constraints, the base of the robot was made from up-cycled parts to create a stable yet flexible stand for the actual continuum robot to stand on. Another change made is that this version contains a four-vertebra prototype with a stronger lower level (higher spring stiffness) and significantly less stiff springs placed between the higher vertebra. This ensures the motors are strong enough to manipulate the entire robot without having to exceed the torque they can generate. Overall, more experimentation with the 3D printing and laser cutting on the parts and different spring stiffnesses (and other physical constants such as motors with more torque) is necessary to perfectly accomplish the final design. However, within


(a) CAD Assembly of the Final Design.

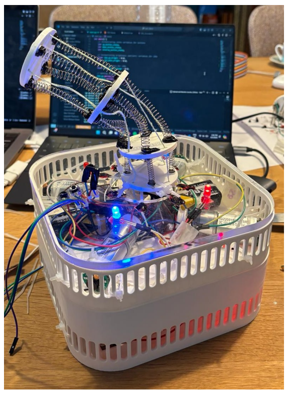

(b) Image of the Built Robot.

Figure 5. Final Assembly, Design and Build.

the scope of this report and project, the built prototype shows the different aspects of the continuum robot and can be built upon or re-constructed further to achieve a closer build.

6 DISCUSSION AND CONCLUSION

In the initial report, we discussed two main tasks we aimed to achieve with our robot: manual control using a joystick or button, and return to neutral state or a designated orientation using PD control. We focused on building the actuation unit and electronics necessary to complete manual control of the robot with five buttons, four of which corresponding to a direction the robot could reorient in and the latter to switch directions when held.

In the process of constructing the revised prototype, we increased the size of the continuum robot to be more proportional to the size of the actuation unit and control system. As we mentioned in the previous Mechatronics and Making term project report, the spring stiffness has a great impact on the overall performance of the robot as it contributes to the stability and compression throughout successive vertebrae. After experimenting with springs of varied stiffness, we chose to use the ones incorporated in the prototype as they were flexible enough to be manipulated by winding pulleys driven by the provided DC motors, as opposed to the springs we used in the previous prototype which were too stiff to be bent via our control system. However, springs with higher stiffness controlled by stronger motors would be better for more accurate control since the effect of gravity (hence deviation from the path or bending in the springs) is reduced as the tension in the strings and the spring stiffness increases, which can be achieved with stronger motors and stiffer springs.

Another change reflected in our control program is the switch from using encoder recordings to control the LED brightness, to manual odometry to keep track of the brightness. By assigning a variable to keep track of the brightness, we were able to adjust the brightness as necessary at each iteration, allowing for a simpler way of controlling the brightness display. Thus, we were able to achieve our objective of displaying circuit output from a simpler angle.

The last change to be noted is the addition of the reverse button which is not shown in the design but can be seen in Figure 5b. The reverse button was the only joint component between the two microcontrollers, connected using a common ground and positive input to display the same state to each, allowing us to reverse all buttons simultaneously. This change was made to allow for each motor to run independently and making winding and unwinding easy for each individual motor.

To conclude, the constructed robot successfully completed most of the design goals with a few changes made due to simplification or lack of resources. Further improvements can be made by reducing the number of wires to allow for better cable management (using a soldering board or another breadboard), adding PD control to perform semi-autonomous tasks and changing out the motors and springs to have better stability.

7 APPENDIX 1: PROGRAM CODE

The program code can be seen below which was run on two separate laptops simultaneously to control the two axes and two microcontrollers separately. The full code can also be found in the following Github repository linked here.

```
#include "mbed.h"
#include <UCL_Encoder.h>

// Motor and Shield Port initializations
PwmOut motor1_PWM(D5);
PwmOut motor2_PWM(D4);

DigitalOut motor1_dir(D6);
DigitalOut motor2_dir(D7);

DigitalOut Enable_Motor_Shield(A0);

float motor1_voltage = 0.0;
float motor2_voltage = 0.0;

// Button initializations
DigitalIn up_button(D0);
DigitalIn down_button(D1);
DigitalIn reverse_button(D3);

// LED initializations
PwmOut led_up(PE_12);
PwmOut led_down(PE_14);

int main() {

    Enable_Motor_Shield.write(1);

    // Initializing motor variables
    motor1_PWM.period_us(50);
    motor2_PWM.period_us(50);

    motor1_dir = 1;
    motor2_dir = 1;

    motor1_voltage = 0.25;
    motor2_voltage = 0.25;

    // Instantiating state variables for buttons
    int up_button_state;
    int down_button_state;
    int reverse_button_state;

    // Instantiating brightness (percent) variables for LEDs
    float led_up_bright = 0.0;
    float led_down_bright = 0.0;

    // Change variable for LEDs
    float brightness_change = 0.0;

    // Setting LED periods
    led_up.period_us(50);
    led_down.period_us(50);

    // Continuous Loop
    while(true) {

        // Collecting input from the buttons
        up_button_state = up_button.read();
        down_button_state = down_button.read();
        reverse_button_state = reverse_button.read();

        // Setting motor directions with respect to the input from the
        // reverse button, while also changing the brightness increment
        // respectively
        if(reverse_button_state == 1) {

            motor1_dir = 0;
            motor2_dir = 0;

            brightness_change = -0.01;

            printf("Button reversed \n");

        } else {

            motor1_dir = 1;
            motor2_dir = 1;

            brightness_change = 0.01;
        }

        // Activating motor 1 if the up button is pressed
        if (up_button_state == 1 && down_button_state == 0) {

            // Setting both motors
            motor1_PWM.write(motor1_voltage);
            motor2_PWM.write(0.0);

            // Changing LED brightness if within range
            if(led_up_bright <= 1 && led_up_bright >= 0) {
                led_up_bright = led_up_bright + brightness_change;
            } else if(led_up_bright > 1) {
                led_up_bright = 1;
            } else if(led_up_bright < 0) {
                led_up_bright = 0;
            }

            printf("Up Button pressed\n");

        // Activating motor 2 if the down button is pressed
        } else if (up_button_state == 0 && down_button_state == 1) {

            // Setting both motors
            motor2_PWM.write(motor2_voltage);
            motor1_PWM.write(0.0);

            // Changing LED brightness if within range
            if(led_down_bright <= 1 && led_down_bright >= 0) {
                led_down_bright = led_down_bright + brightness_change;
            } else if(led_down_bright > 1) {
                led_down_bright = 1;
            } else if(led_down_bright < 0) {
                led_down_bright = 0;
            }

            printf("Down Button pressed\n");

        // Activating both motors when both are pressed for compression
        } else if (up_button_state == 1 && down_button_state == 1) {

            // Setting both motors
            motor1_PWM.write(motor1_voltage);
            motor2_PWM.write(motor2_voltage);

            // Changing LED brightnesses if within range
            // UP LED
            if(led_up_bright <= 1 && led_up_bright >= 0) {
                led_up_bright = led_up_bright + brightness_change;
            } else if(led_up_bright > 1) {
                led_up_bright = 1;
            } else if(led_up_bright < 0) {
                led_up_bright = 0;
            }

            // DOWN LED
            if(led_down_bright <= 1 && led_down_bright >= 0) {
                led_down_bright = led_down_bright + brightness_change;
            } else if(led_down_bright > 1) {
                led_down_bright = 1;
            } else if(led_down_bright < 0) {
                led_down_bright = 0;
            }

            printf("Both buttons pressed\n");

        // Setting both motors to zero when nothing is pressed
        } else {
            motor1_PWM.write(0.0);
            motor2_PWM.write(0.0);
        }

        // Outputting the allocated brightnesses for the LEDs
        led_up.write(led_up_bright);
        led_down.write(led_down_bright);

        thread_sleep_for(100);

    }

    // Turning off the motos and LEDs
    motor1_PWM.write(0.0);
    motor2_PWM.write(0.0);

    led_up.write(0.0);
    led_down.write(0.0);

    thread_sleep_for(200);
}
```

Listing 1. Final Project Code

8 APPENDIX 2: VIDEO

The link to the associated video of this project explaining the technical and design aspects can be found linked here.

REFERENCES

[1] Doroudchi, A. and Berman, S. (2021). Configuration tracking for soft continuum robotic arms using inverse dynamic control of a cosserat rod model. In *2021 IEEE 4th International Conference on Soft Robotics (RoboSoft)*, pages 207–214.

[2] Ghoul, A., Kara, K., Mohamed, B., and Boualem, N. (2021). Control of continuum robot using two optimized pid controllers.

[3] Rao, P., Peyron, Q., Lilge, S., and Burgner-Kahrs, J. (2021). How to model tendon-driven continuum robots and benchmark modelling performance. *Frontiers in Robotics and AI*, 7.

[4] Russo, M., Sadati, S. M. H., Dong, X., Mohammad, A., Walker, I. D., Bergeles, C., Xu, K., and Axinte, D. A. (2023). Continuum robots: An overview. *Advanced Intelligent Systems*, 5(5):2200367.

[5] Sincak, P. J., Prada, E., Miková, , Mykhailyshyn, R., Varga, M., Merva, T., and Virgala, I. (2024). Sensing of continuum robots: A review. *Sensors*, 24(4).

[6] Wei, D., Zhou, J., Zhu, Y., Ma, J., and Ma, S. (2023). Axis-space framework for cable-driven soft continuum robot control via reinforcement learning. *Communications Engineering*, 2(1).