EPFL

Semester Project

Semester of Winter 2024

Treadmill with body weight support robot implementation

Marianne Civit Ardevol (325056)

Supervisor : Victor Perez Puchalt
Laboratory : G-lab, Gregoire Courtine

January 10, 2025

Contents

1 Introduction

The assessment of motor function rehabilitation plays a crucial role in the development of technologies for spinal cord injuries and rehabilitation processes. These technologies often rely on preclinical models to study rehabilitation processes, understand recovery mechanisms, and test interventions. The use of a robotic platform based on a treadmill with a body weight compensation system is commonly used for rodents and has demonstrated its utility in facilitating motor recovery studies [2][3]. This type of system allows injured and recovering rodents to engage in controlled walking exercises whilst not applying all of their weight on all limbs as well as the possibility of walking in a biped manner.

Alexandre Lechartier, a former student, developed this type of robotic system at the G-lab to address the limited availability of easily accessible treadmills for research purposes. The developed robot has a treadmill, body weight support system and reaction force measurement plates. Due to timing constraints, the software was not fully finished and needed to be completed.

This semester project aims at advancing the created robotic platform in order to obtain a functional treadmill with body weight support compensation. The main focus of this work is mainly the software implementation since all of the hardware was already designed. The report will go through the multiple steps and techniques used for achieving this functional system. Four aspects were mainly developed and carried out during this project:

1. Resolving the communication between different modules
2. Adding a calibration procedure for the body weight support system
3. Implementing the body weight compensation
4. Fixing and replacing hardware components

These points will be developed in detail in this report. An additional informative document will be destined to the next student working on the robot and will provide the main information required for easily taking over and a final document will provide a user manual.

Figure 1: Image of the G-mill

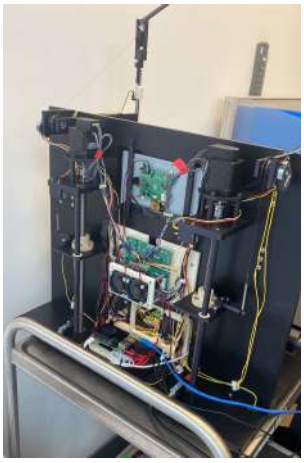


Figure 2: Inside the G-mill

2 State-of-the-art

2.1 Existing treadmills with body weight support

The use of treadmills with body weight support (BWS) for rehabilitation of patients with lower extremity motor function deficit have already been implemented [1]. These treadmills allow humans to undergo efficient recovery and allow for walking without applying too much force or pressure on the weak limbs as seen in figure 3.
To define accurate recovery models, laboratories use rodents to observe and obtain the best recovery process and translate the knowledge to human recovery. Therefore such robotic platforms are also useful in laboratories and need to be adapted to satisfy the morphology, size and weight differences of rodents and humans. As discussed in the introduction, these robots can be used for many other purposes such as gait analysis and techonology testing.

Figure 3: Treadmill with BWS for humans [4]

Figure 4: Treadmill with BWS for rodents [5]

The main robotic platform used for rodents with a BWS is the robomedica robot, this robot is also used at the G-lab but its production has stopped and it is now difficult to replace pieces and parts. The existing robomedica robot was the main source of inspiration for the treadmill design of the previous student.

2.2 Existing treadmill designed by previous student

Alexandre Lechartier, a previous student has developed the current treadmill and its hardware. His work was inspired by the robomedica robots but everything was re-designed and implemented from scratch. This robot consists of two treadmills, one side for mice (left) and one side for rats (right) with their respective body weight support systems as seen in figure 1. These systems are made of a linear pulley and spring to compensate the desired force and alleviate the weight of the rodent.

A few aspects need to be explained in order to continue the description of this project. Furthermore, part of this projects work was to understand what had been done and how to advance it. Therefore, the following part will briefly explain the main concepts and modules of the robot that dictate its behaviour (for more information, please refer to the Alexandre's report).
The system is composed of two microcontrollers: an arduino due and a raspberry pi.

Figure 5: Arduino location in the robot

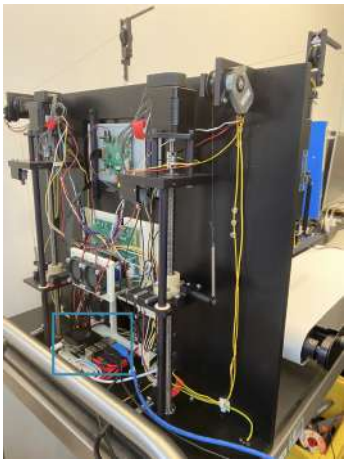


Figure 6: Raspberry location in the robot

Arduino: The arduino is placed inside the core of the robot, in between the treadmills as shown in image 5.
This controller will dictate the behaviour of the treadmill motors, BWS motors and receive input when the endswitches are pressed. It controls all of the hardware of the robot. The code is written in C++ with a different file for every execution task type and interrupt service routines when endswitches are pressed. There is a total of 8 endswitches : 4 "internal" switches placed inside the back of the robot (in figure 6), to indicate the end of route of the lead screw and 4 "external" switches to indicate the end of route of the BWS 3D printed piece where the rodent is placed. Furthermore, a rotational encoder and button, placed on the front of the robot allows the user to interact with the treadmill.

Raspberry pi: The raspberry pi is placed in the back of the robot as shown in image 6 and it is linked to a GUI.
Its goal is to receive the main information input from the user through the interactive touchscreen and communicate with the arduino to adjust the robot's behaviour respectively. The raspberry pi is linked to the screen where the raspberry IDE can be found. All of the code is found there and can be modified via ethernet connection. Indeed, the raspberry pi is connected via serial communication to the arduino and is used to flash code onto it. Once this is done, the raspberry's code can be run to make the robot function. It is programmed in python and mostly runs the GUI interface and communication to the arduino.

The current state of the robot when starting this project was the following :

- Most of the hardware was functional
- The 3D printed body weight support of the mice needed replacement
- The mice encoder needed to be fixed
- The treadmill could work with bugs and some communication issues that needed to be fixed
- The body weight support code was not fully implemented

The prior work accomplished was exceptional, and this project seeks to build upon it by further advancing the software.

3 Methods

3.1 FSM states and procedure

This section outlines the main states of the machine as well as its operating procedure. These will be further developed in the following parts.

The main robot's implementation code is placed on the arduino since it controls its hardware. The raspberry pi only serves as an interface between the user and arduino for communication. Therefore most of the code is found on the `main.cpp` file of the arduino. This file takes care of setting up all of the hardware and continuously updating their state. The operational process is as follows :

1. Check if the internal and external endswitches have been pressed and modify the needed variables
2. Control of the treadmill
3. Control of the lead screw's motor and therefore of the body weight support system
4. Check the rotational button and act accordingly
5. Communicate with the raspberry pi

The core of the robot's function is within the control of the treadmill and lead screw for the BWS motors. With the aim to follow a systematic order, stay consistent in the tasks and distinguish them, the robot will take different states. By going through them, the procedure can be analysed and understood. Figure 7 shows how the different states interact and how the machine goes from one state to another. The blue boxes show each state, the grey boxes show input from the GUI, and the text show events within the arduino's hardware inputs.

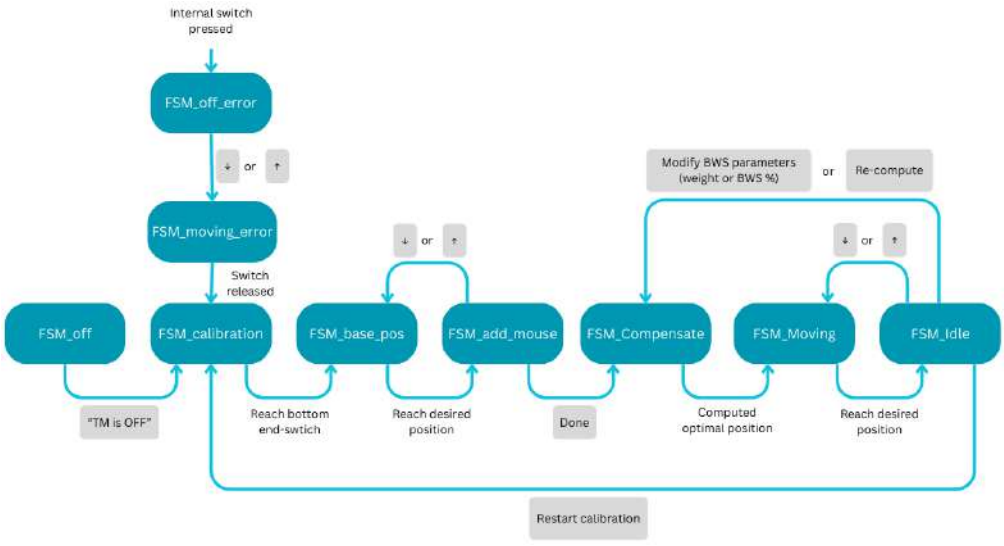


Figure 7: States of the robot

First, the robot is in its `FSM_off` state. In this state, the treadmill and BWS cannot be actuated, the GUI is activated but no hardware will be able to move even if the speed variable is increased. This ensures a safe state for the machine to be in when working with rodents. This state can be

promptly activated at any time by pressing the round rotational button or selecting the "TM is ON" option on the user GUI. The button will then show "TM is OFF", indicating that the machine now off.

When the robot is turned on by pressing the "TM is OFF" button (thus becoming "TM is ON"), the machine goes into its `FSM_calibration` mode to calibrate the motors and ensure a reliable reading of the encoders. It then goes onto the `FSM_base_pos` to move into its base position and wait for the rodent to be added in `FSM_add_mouse`. In this state, the user has the option to either directly secure the rodent onto the 3D printed support or press the up ↑ and down ↓ arrows to make this support move and be in a more ideal position for placing the rodent. Once the rodent is secured, the user must insert the weight and desired BWS compensation and finally press the "done" button. The machine will go through the three states `FSM_Compensate` to compute the position needed for the inserted weight and BWS compensation, `FSM_Moving` to move the the computed position and `FSM_Idle` once finished.
In this state, the user has the option to modify the BWS parameters or re-compute the position for the ones inserted if they believe that it is not accurate. Furthermore, the base position of the 3D support can be asjusted manually with the arrows.
Finally, an additional part of the loop takes into account the unlikely event that an internal switch is pressed going into `FSM_off_error`, therefore stopping the robot and needing the user to press the arrows to manually unblock it. The robot will restart its calibration once the switch is released.

Furthermore, when the robot is turned off, its current state will be saved in a variable allowing it to return to its previous state when turning it back on. This feature enables the machine to maintain dynamism and eliminates the need to restart the calibration process each time the robot is turned off during a session.

3.2 Calibration method

For the robot to accurately perform a body weight compensation, the system must be calibrated at each start of the machine. In order to understand how the calibration works, an explanation of the system must first be made.
The way the body weight support works is the following : a motor actuates a lead screw to which a platform is attached. On top of it, a spring is placed and linked to the 3D printed rodent support with a string passing through a pulley as shown in figure 8.

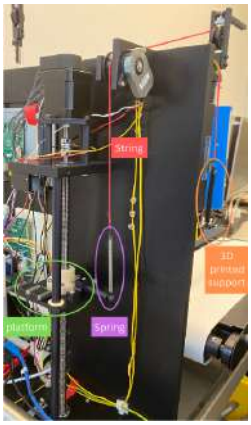


Figure 8: Calibration components

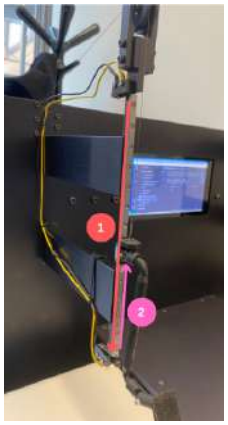


Figure 9: Calibration trajectory

Finally, the movement of the platform regulates the spring tension as well as the position of the 3D printed support. An encoder is placed on the pulley in order to know the position of the end-piece. Therefore two measures are recorded : the position of the motor (i.e position of the platform and bottom of the spring) and the position of the pulley (i.e. the position of the 3D printed support and top part of the spring).
Since this support has a limited range of motion and its position needs to be exactly known, a calibration must be done. What is here meant by calibration, is that the position of the motor encoder as well as the one of the pulley encoder are both reset when reaching the beginning of the range of motion as seen in figure 9, when the support reaches the end of trajectory 1.

Therefore, a code was implemented on the arduino's controller in the `weightsupports.cpp` file when it is in the `FSM_calibration` state. Upon startup, the lead screw's motor will be actuated to bring the platform all the way up, therefore bringing the rodent support all of the way down. The motor will continue until the support presses the bottom external endswitch (see figure 9). Once it is pressed, the switch variable changes to indicate the end of range and two things happen:

1. The motor changes its rotation direction, therefore bringing the support back up to its base position. This position was determined empirically by observing the heights of mice and rats, ensuring that the support aligns with their respective heights.

2. The positions of the lead screw's motor and pulley encoder are reset.

The position of the lead screw's motor is set to zero since this is the least it can go and we assume it is its origin. For the pulley's encoder position, a different value is used. The pulley's encoder represents how much the spring is elongated (see section 3.4) and should ideally be zero when the switch is pressed since the 3D support rests on it. Nevertheless, when the switch is actuated, the time to change motor direction, means that the spring may be a little bit more compressed than its zero position. The pulley's encoder position are therefore set to $-11mm$ and $-20mm$ for the rats and mice respectively. These values were set by measuring the spring elongations and comparing them to their base value as well as by iterative testing of the robot with weights.

3.3 Progressive acceleration

The treadmill's motor is a stepper motor, therefore its control is done by sending a series of pulses proportional to the desired speed. In the code, a function computes the needed time between each pulse and then sends the information only if this time has elapsed to regulate the speed.
One issue with this code was the problem that when abruptly switching speeds with a large difference, the change would make the motor very noisy or even sometimes make the robot reboot because a lot of power was needed. To remediate this issue, a *progressive acceleration* was put into place. Indeed, two variables were introduced: `actual_speed` and `desired_speed`. With these, the actual speed was progressively increased until it reached the desired speed, ensuring a safe change. A first attempt of the progressive increase function is shown in figure 11 where v_a is the actual speed of the treadmill, v_d is the desired speed of the treadmill and v is the next speed.

However, this function is not optimal and may sometimes not work when trying to decrease the speed if $0.25 \cdot (v_d - v_a) < 1$. After careful consideration, it was decided that a linear function would be the most suitable choice. The final function to be used is shown in figure 13.

This function adjusts the speed in increments of 0.5 linearly and ensures a stable speed acceleration. When the desired speed is at 1m/s from the actual one, then the desired speed is directly set.

$$v = \begin{cases} v_a + \frac{1}{1+0.25\cdot(v_d - v_a)} & \text{if } |v_d - v_a| > 1, \\ v_d & \text{if } 0 \leq |v_d - v_a| \leq 1 \end{cases}$$

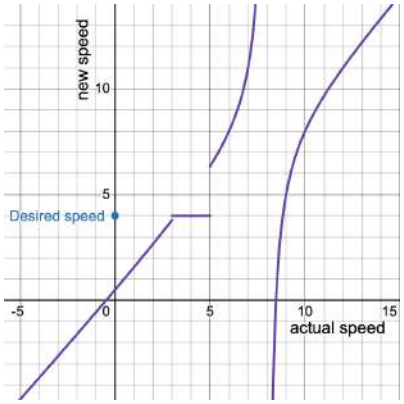


Figure 10: Function for desired speed of 4

Figure 11: Non-linear progressive speed acceleration

$$v = \begin{cases} v_a + 0.5 & \text{if } v_d - v_a > 1, \\ v_a - 0.5 & \text{if } v_d - v_a < -1, \\ v_d & \text{if } 0 \leq |v_d - v_a| \leq 1 \end{cases}$$

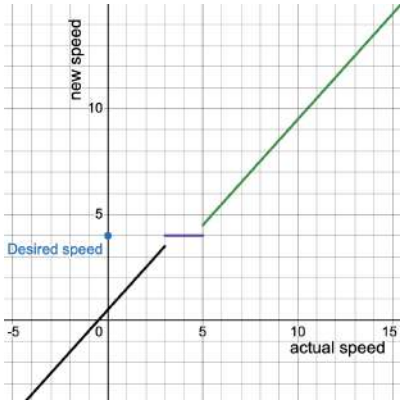


Figure 12: Function for desired speed of 4

Figure 13: Linear progressive speed acceleration

3.4 Body weight support system

An important feature of the robot is its ability to compensate the weight of the rodent through its body weight support system. The way this is done is by using the reaction force of a spring to compensate for the weight force of a rodent. This idea came from Alexandre Lechartier and can be found on his report. For convenience purposes, it will be explained again with additional descriptions of what was understood during the project as well as how it was implemented by code.

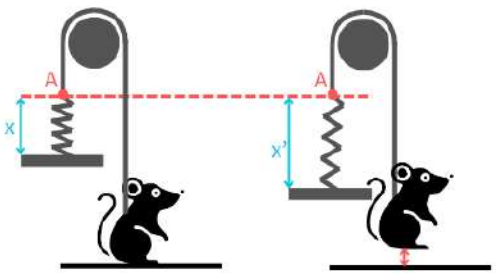


Figure 14: Pulley and BWS robot system

Figure 14 shows a graph of the system where a mouse's weight can be supported by an elastic spring. Indeed, by moving the platform downwards, the top point of the spring (point A) should not move such that the spring elongates, therefore applying a larger elastic force and compensating for the rodent's weight.
The formula for the force used is the following:

$$F_{rodent} = (w_r * (\text{BWS} * 0.01) + w_s) * g * 0.001$$
$$F_{spring} = k * \Delta x + T_{init}$$

Where w_r is the weight of the rodent in grams, BWS is the percentage of body weight support needed, w_s is the weight of the 3D printed rodent support and $g = 9.81$ is the gravitational constant. Furthermore, k is the spring's constant, Δx is the spring's elongation and T_{init} is the initial tension of the spring. Therefore, the needed spring elongation such that the spring's force compensates for the desired rodent's force can be deducted when $F_{spring} = F_{rodent}$:

$$\Delta x = \frac{F_{rodent} - T_{init}}{k} = \frac{(w_r * (\text{BWS} * 0.01) + w_s) * g * 0.001 - T_{init}}{k}$$

Regarding its implementation on the robot, this needed elongation is compared to the actual spring's elongation to find the required displacement and converted to motor signals to move the platform accordingly.
In order to determine the current spring's elongation, the encoders come into play. Indeed, the motor encoder will reflect the platform's position whereas the pulley's encoder will determine the position of the top of the spring (point A). Therefore, by subtracting the position of the motor's encoder to the one of the pulley's encoder, the spring elongation is found $\Delta x = x_{mm}^{platform} - x_{mm}^{A}$ since the initial elongation of the spring is taken into account when resetting the position.

For coding purposes, all of the calculations are done in mm in the `weightsupports.cpp` file and then converted to motor or pulley signals to actuate them. A full rotation of the lead screw's motor represents 500 motor signals and a linear displacement of 2mm, therefore the position of the platform can be extracted in mm and the needed displacement can be reconverted in motor signals:

$$x_{mm} = \frac{x_{signals} * 2}{500} = \frac{x_{signals}}{250} \qquad \text{to get the position of the platform}$$
$$x_{signals} = \frac{x_{mm} * 500}{2} = x_{mm} * 250 \qquad \text{to get the signals needed to move by x mm}$$

For the pulley's encoder, only the position needs to be extracted. The pulley's resolutions are different in the mouse's and rat's side due to the way they were mounted, however this is not necessary. Their resolution is 5120 and 1000 signals for one rotation respectively and the pulley's diameter is $24,5$. Therefore to get the position of point A from the pulley's encoder, the following formula is used:

$$x_{mm}^{A} = \frac{x_{signals}^{A} * 2\pi * d/2}{\text{resolution}} = \frac{x_{signals}^{A} * pi * d}{\text{resolution}}$$

Finally, the spring's elongation can be found $\Delta x = x_{mm}^{platform} - x_{mm}^{A} = \frac{x_{signals}^{platform}}{250} - \frac{x_{signals}^{A} * pi * d}{\text{resolution}}$

These computations are made and allow the system to compensate for the rodent's inserted weight and BWS percentage. Furthermore, to make sure that the rodent is well above the treadmill when the BWS is at 100%, an additional 1cm is added to the platform movement.

3.5 Motor actuation and displacement of the platform

To achieve the desired displacement of the platform on the lead screw, a movement strategy is put into place and a a few safety measures must be undertaken.

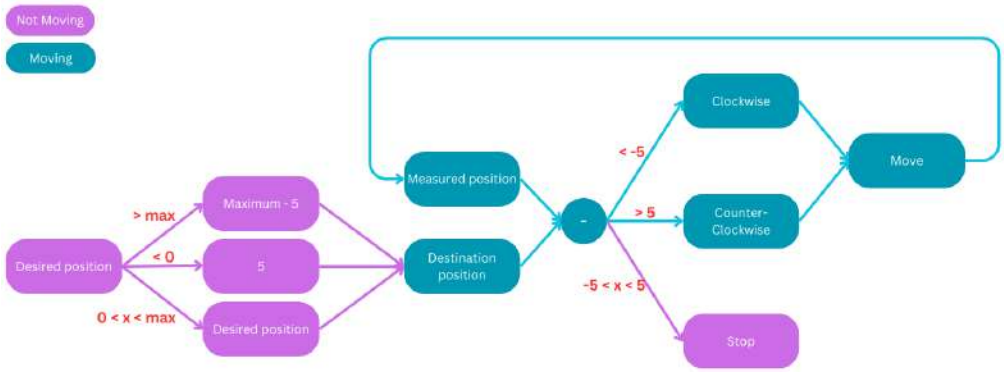


Figure 15: Motor actuation process

First of all, the displacement is dictated by the measured position compared to the desired position of the motor's encoder. If the difference is positive, the rotation direction should be clockwise, if not, it should be counter-clockwise. Therefore by comparing these and adjusting the rotation direction the desired position can be reached. The stopping criterion is that their difference should be less or equal to 5 motor signals (corresponding to $0,02$ mm).
The movement is also mediated by the state in which the robot is in, it will only move if it is in a moving state such as `FSM_Moving` and will change states when reaching destination.

An additional safety measure is added when changing the destination position. Indeed, before setting a new destination, its value must be checked. It will be compared with the maximum and minimum range of the lead screw in case it were to be greater.

- If the position is greater than the maximum : set the destination to the maximum - 5
- If the position is smaller than zero : Set the destination to 5
- If not set it to the desired position

Similar to the treadmill motor, pulses are sent at an interval to define the speed. The speed for the lead screw is constant and is set by sending the bit 1 to the correct pin of the motor for the direction as well as actuation.

3.6 Communication between the Arduino and Raspberry pi

An important aspect of the robot is the communication between the Arduino and Raspberry pi to ensure a reliable and accurate message transmission between the user and robot as well as between the robot and user. The communication is done via serial connection at a baud rate of 115200.

As explained above the raspberry pi mostly serves to show the user interface and provide information to the arduino regarding the treadmill speed, BWS %, rodent weight and machine states (on, off, calibration). In contrast, the arduino communicates its important information such as the state of the machine (has it finished calibration...), the change in speed with the rotational button and error messages from the hardware.

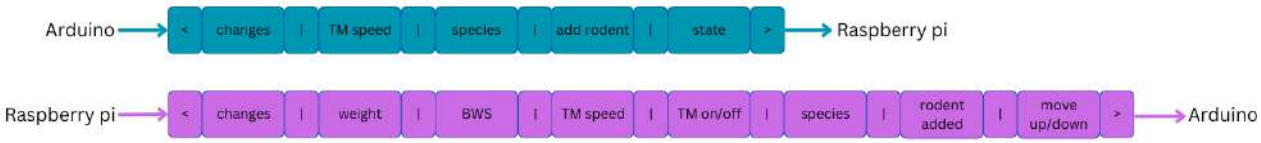


Figure 16: Communication process

Figure 16 shows the data format sent by the arduino and raspberry pi respectively. To keep consistent format and prevent reading errors, each message is sent between a start and end bit ("<" and ">") and each part is separated with a "|" within the message. Furthermore, the messages begin with a set of bits called "changes" as seen in figure 16. This register contains at least n bits, where n is the number of elements sent (i.e. the arduino sends the TM speed, species, add rodent and state, therefore n = 4). Each bit will indicate whether the value sent has changed from the previous message and allows for a more optimal reading. When a "0" bit is in the changes register, the corresponding value will not be read or saved by the receiver. Furthermore, this register is also used for sending information such as restarting the calibration where only a bit is needed and no information regarding the value of this bit is important.

The important values sent by the arduino are the following:

- TM speed : the treadmill speed can be changed with the rotational button linked to the arduino. The value for the speed should be updated on the user GUI.
- add rodent : When the BWS system has finished calibrating, it will send information letting the user know that the calibration is done and that the rodent can now be added.
- state : As seen in section 3.1, the states will update after each task. These should be reflected by the display on the GUI.

The important values sent by the raspberry pi are the following:

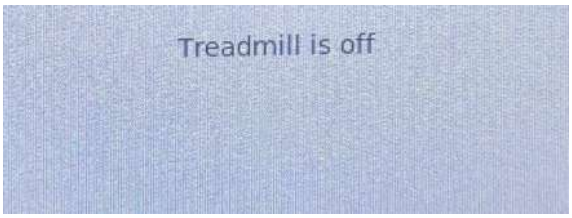
- weight : When the user inserts or changes a value for the weight of the rodent, the information must be communicated to the arduino for accurate BWS.
- BWS : Same as the weight, this value is needed to compute the required spring elongation by the arduino.
- TM speed : the treadmill speed can be changed in the GUI and should be communicated to update the speed by the arduino.
- TM on/off : The treadmill can be switched on or off "TM is ON" or "TM is OFF" in the GUI. This is sent to the arduino to update the state and save the previous state if needed.
- species : The selected species on the GUI is sent to the arduino to enable the correct side of the robot.
- rodent added : When the user has finished securing the rodent on the 3D support and presses the button "Done", the information in given to the arduino such that it can now move on to the `FSM_Compensate` state.
- move up/down : The user has the possibility to manually move the 3D printed support up or down with the arrows. This is sent to the arduino to perform the movement.

3.7 User interface

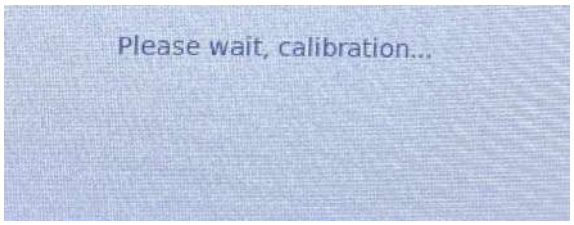
The user interface allows the user to modify the variables and interact with the robot.

Figure 17: User interface at startup

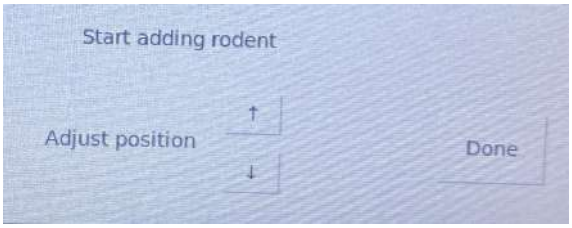
Figure 17 shows an image of the GUI when the robot is first turned on. Five main regions can be observed, the speed, weight and BWS can be adjusted with the buttons or keyboard. Then, the top right corner allows the user to turn the robot ON or OFF and restart the calibration if needed. Finally, the bottom part of the GUI is reserved for notifying the user about the state of the machine and interactively updates.



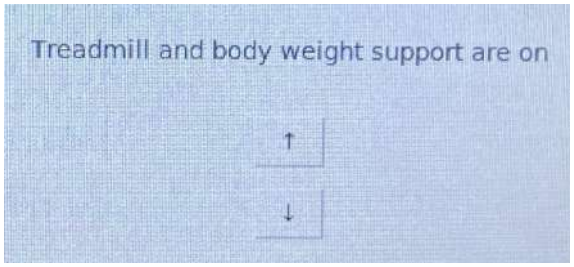
(a) Treadmill off display



(b) Calibration display



(c) Adding rodent display



(d) Treadmill on display

Figure 18: Displays for the lower part of the GUI

There are four possible display options as seen in figure 18. These notify the user when the state changes. The arrows seen in figures 18c and 18d, allow the user to manually move the 3D printed

rodent support upwards or downwards.
All of the code for the GUI is implemented on the raspberry pi with the tkinter module.

3.8 Fixing and replacing hardware components

Pulley encoder: First of all, the mice's pulley encoder was not correctly working. Two hypotheses were made: either the encoder was broken or the connections were weak. It was found that the problem came from the connections. Indeed, as seen in figure 19 the encoder was connected with a custom pcb and the cables were preventing the connectors to fully reach the encoder.

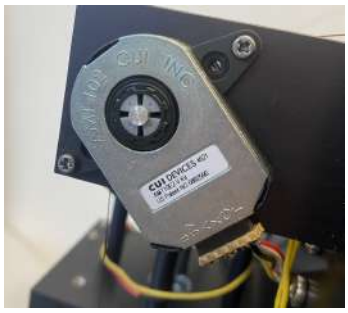


Figure 19: Pulley encoder

Figure 20: Mice end-switch

To fix this, header cables were used instead. Therefore, they were soldered to the current cables in order to keep the correct connection and fix the encoder's stability. Furthermore, it was found that the set resolution on the mice's encoder was higher than the one on the rat's encoder, meaning that it is much more sensible to small movements and noise.

External endswitches: Another hardware adjustment made was changing the external endswitches on the mice's side (see Figure 20). Alexandre Lechartier, the previous student had ordered new switches requiring less force to be actuated and therefore making the calibration easier. Since the 3D printed mice support is smaller and weighs less than the rat's one, the force that it creates when resting on the end-switch was not enough to efficiently actuate it. These were then changed to switches requiring a force of 6g to be actuated and allowed them to be easily switched.

3D printed rodent support: Finally, an adjustment observed when testing the robot with real mice was the design of the 3D printed rodent support.

Figure 21: Previous endpiece

Figure 22: New 3D rodent support

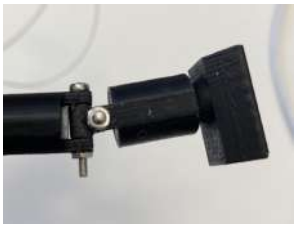


Figure 23: New endpiece

An observation made by the users was that it could be very interesting to have an additional degree of freedom on the end-piece. The support was previously able to only move up and down but not side to side as seen on figure 21. Since the mice are secured with a velcro harness, it tends

to not be exactly centered on the support meaning the mice may be tilted towards a side.

In order to achieve such a movement, a 3D piece was designed and inspired from a universal joint as seen on figure 23. This allows the user to get two degrees of freedom and freely move the rodent once it is secured on the support. The use of screws and bolts allow the pieces to be more or less tightened and allow to have a stiffer or looser connection and movement.

External power source: Since the built-in power source is not reliable and creates power or current peaks randomly, burning the electronics, an external power source was used. The power source is a voltage generator set at $24V$ and a maximum of $2mA$. Since this source is connected via cables from the inside of the robot, the hole dedicated to a switch was used for passing the cables through. Indeed, a hole was originally created for holding a switch which would allow to turn on/off the machine. However, since an external power source needs to be used, this hole was re-designed to let the cables go through. In order to do this, the power source crocodile cables were soldered to smaller ones such that they could be fastened to the arduino's power input module. Finally an additional 3D piece was printed to enhance the design of the robot and increase its safety by having smaller holes to the inside electronics.

Figure 24: External power supply

Figure 25: Hole for cables

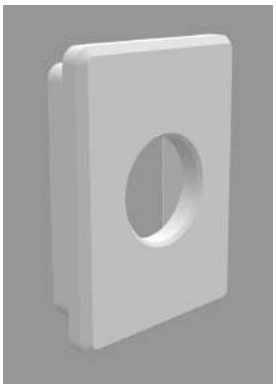


Figure 26: 3D designed hole

Noise reduction: Another issue that needed to be solved was the noise created by the actuation of the lead screw. Since the platform is linked to the lead screw, some movement is translated and creates friction with the rod seen in figure 27. Furthermore, and most importantly, the presence of the additional piece for holding the spring adds weight on one side and creates an uneven weight distribution, adding more friction.
To solve this, additional weights were placed on the opposite side of the platform to even out the weight distribution. Indeed, a scrap piece of steel metal was found in the EPFL's skill laboratory. Its weight was $138g$ for a volume of 80x40x5=18'170mm^3 giving it a density of $\approx 7.6g/cm^3$. To determine the needed weight to compensate for the noise, the use of multiple large bolts were placed on the platform to test the result and then weighed. A weight of $60g$ was determined to be acceptable for the rats side and $30g$ for the mice's side. Therefore, taking into account spacing constraints, the sizes were found: two plates of 5x12x65mm^3 were used for the rat's side and the remaining metal was used for the mice. These were then superglued together with an additional layer of foam to absorb the vibrations and then taped to the platform for additional security.

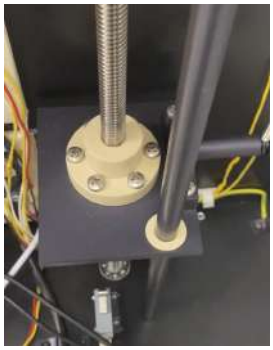


Figure 27: Noisy platform without weights

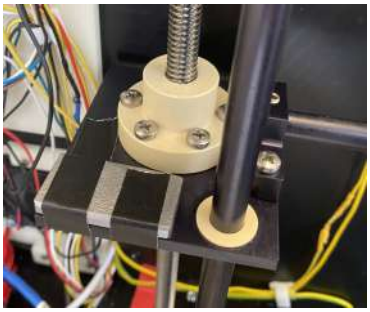


Figure 28: Additional weight on mice's side

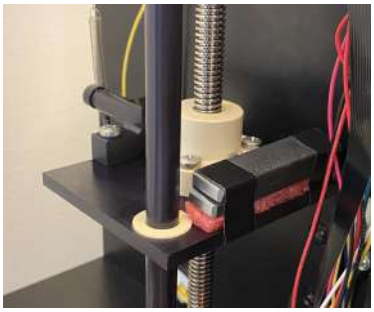


Figure 29: Additional weights on rat's side

3.9 End position error handling

A final implementation done during the project was the handling of the unlikely event of the system reaching its end position. As mentioned above, the system is created such that the computed position of the platform is constrained between the maximum and minimum values of the range of motion. However, the case where this is not satisfied must be considered in case the code were to fail or an error were to arise, causing the system to go over its limit range of motion. In this situation, the system must be completely stopped to prevent damage to the robot and the user must be notified of the problem.

To prevent the system from going any further, the internal end-switches are used to indicate the end of range. Therefore, once the platform reaches one of these internal switches, an interrupt service routine directly switches the robot off and puts it into its `FSM_off_error` state as shown in figure 7 from a previous section. This allows the system to remain safe.

A message is then communicated from the arduino to the raspberry pi to indicate the error and update the GUI for the user to be able to interact with it. The user will receive a message: "The Body Weight Support system has reached its limit. Please adjust its position with the arrows to resolve the blockage" along with a button containing the correct arrow for making the platform move in the opposite direction and unblocking the system. Once the endswitch is released, the system will undergo a new calibration and restart the entire process.

This part of the system was tested a few times, however due to timing constraints it was unfortunately not optimised. The system works well in the case that the top switch is pressed, however in some cases, when the bottom switch is pressed, the arrow button to unblock the system correctly indicates the movement direction but the movement is opposite to the one needed. This means that the platform will try to go closer to the endswitch rather than away from it. This problem should be fixed in the near future.

4 Results

This project enhanced the main functionality of the robot under different aspects. The treadmill function and the communication between the two microcontrollers was improved and allowed for a functional treadmill and body weight support system.

To quantify and demonstrate the accuracy of the body weight support system, multiple tests were done with different objects of different weights. Indeed, by securing objects of known weights to the rodent support piece and using a scale to observe how its ground reaction force varied, the BWS system was evaluated. The table below shows multiple measures performed on different objects and the results for each support percentage.

Actual weight (g)	11.6		15.4		18		19	
Inserted weight (g)	11		16		18		19	
Support (%)	Theoretical weight (g)	Measured weight in average (g)	Theoretical weight (g)	Measured weight in average (g)	Theoretical weight (g)	Measured weight in average (g)	Theoretical weight (g)	Measured weight in average (g)
0	11.6	11.50	15.4	13	18	18.2	19	18.47
10	10.5	10.90	13.8	13	16.2	18.1	17.1	18.20
20	9.4	9.60	12.2	12	14.4	17.15	15.2	17.07
30	8.3	8.83	10.6	10	12.6	14.8	13.3	15.37
40	7.2	7.40	9	8	10.8	13.35	11.4	13.23
50	6.1	6.47	7.4	6	9	12.25	9.5	11.27
60	5	5.37	5.8	6	7.2	11.4	7.6	9.40
70	3.9	4.20	4.2	6	5.4	9.1	5.7	7.37
80	2.8	3.10	2.6	4	3.6	7.5	3.8	5.20
90	1.7	2.13	1	2	1.8	4.2	1.9	3.63
100	0.6	1.50	-0.6	0	0	0	0	1.30

Figure 30: Results table

The actual weight represents the true weight of the object used whereas the inserted weight is the weight inserted as input on the GUI that was then fed to the computations of BWS by the arduino for compensation. The theoretical weight represents what the scale should be reading with the actual and inserted weights and support percentage. It is found with through this computation:

$$w_{theoretical} = w_{actual} - (w_{inserted} * \text{support}\% * 0.01)$$

This test was mostly performed on the mice's side since it is the one that will currently be most used at the lab and the one that contains the treadmill belt. It can be seen that most measurements are accurate and only vary slightly. However, during measurements, the measures tended to vary from time to time. A few additional measures show its accuracy:

- The mean absolute error of the system shows a measure of the accuracy and is $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| = 1.4$, which is acceptable.
- The mean standard deviation of the measurements shows how consistent the system is $\overline{\sigma} = \frac{1}{n}\sum_{i=1}^{n}\sigma_n$ where $\sigma_n = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_{i,n} - \overline{\hat{y}}_n)^2}$, it is found to be $\overline{\sigma} = 1.28$

An additional aspect observed during the tests is the presence of static friction or hysteresis in the spring. Indeed, when testing the weights and adding support %, if the system was left untouched, it would show a certain weight on the scale. However if the user touches the 3d support such as to lift it and let it go down on its own, the weight measured is different and more accurate. This indicates that the spring may have been stuck to its previous position since the movement corresponding to

a 10% difference of support is relatively small.

Finally, to evaluate the treadmill's performance and identify potential areas for improvement, it was tested with mice midway through the project. This early testing session provided insights into both the system's strengths and the aspects needing improvement. During the session, several aspects were noted to function effectively, while others showed opportunities for improvement.
As detailed in the previous section, these observations led to updates, including the development of a new 3D rodent support system with two degrees of freedom, which enhanced the adaptability and comfort for the rodents. Additionally, improvements were made to the body weight support system code, ensuring better functionality and reliability during operation.
A follow-up testing session with mice was planned to validate these enhancements before the project's conclusion. Unfortunately, due to logistical constraints, this second session was unfeasible. In its place, a test using calibrated weights was conducted, which provided sufficient data to assess the system's performance and confirm the effectiveness of the implemented improvements.

Throughout the course of the project, the system underwent repeated testing and analysis to increase its functionality and enhance its overall effectiveness. Each test provided feedback, allowing incremental improvements and ensuring that all components worked seamlessly together. The new elements integrated to the robot during the project should function reliably, laying the groundwork for future enhancements. Additional features can be incorporated into the robot easily, maintaining flexibility and scalability for subsequent developments.

5 Discussion

During the course of this project, significant progress was made in refining the G-mill robot, ensuring its readiness for initial use in laboratory experiments involving rodents. One of the primary advancements was the improvement of the calibration process, which now ensures accurate and reliable operation during tests. The body weight support system was carefully optimized to provide stable support for the rodents, a crucial feature for the robot's intended applications. Additionally, the treadmill actuation was fine-tuned, enabling smooth and controlled motion to meet the requirements of experimental protocols. The graphical user interface (GUI) was also improved, allowing it to process user inputs more effectively while maintaining ease of use. These collective enhancements have resulted in a dependable and versatile robot, setting the stage for future improvements and additional features.

Even though the robot is now functional, additional work should be done to fully expand its potential. Indeed, the robot was created by Alexandre Lechartier with the idea of being able to record the ground reaction force exerted by each limb of the rodent. Therefore, piezoelectric force plates were included below the treadmill belts such that the rodents' limbs would actuate it when walking or running. This would bring a lot of valuable insight into the recovery process of rodents. However, this complex part was not implemented in the current project due to time constraints but should be implemented in the future.

Regarding the body weight support system, section 4 shows that it is accurate enough for non-specific and general use of the robot. However, a few challenges have arisen with the use of springs in the body weight compensation system. As mentioned, due to their reactivity and sensitivity to small movements, the springs may tend to stick to a position when small movements are needed, which therefore cause a decrease in accuracy. If the BWS system were to be used for extremely precise research, it should be re-calibrated by adjusting the spring constants as well as the reset value of the pulley's encoder and re-tested. Regarding, the rat's side, tests were done but a re-calibration of the constants is always beneficial before beginning to use it.

Such a project comes with opportunities for improvement. Based on the insights gained throughout the current project and the feedback received during testing sessions with mice, a comprehensive list of potential enhancements was developed. These improvements aim to address observed challenges, ameliorate existing functionalities, and expand the robot's capabilities for future applications.

- Implement the ground reaction force measurement and show a graph of the recorded values on the GUI.
- Fix or replace the integrated power source such that it can be safely used for the robot.
- Change the stepper motor to a DC motor such that less noise is generated and less power is consumed, allowing the treadmill to go faster.
- Look into the problem where the robot suddenly reboots. This has happened a few times without a clear reason, the assumption is that the power consumption might have peaks and cause a reboot of the machine.
- Finish the implementation of the error when the platform reaches an internal end-switch as described in section 3.9.

- Order new treadmill belts. Emails have been exchanged with the french company vulca concept (`https://www.vulca-concept.com`), confirming the possibility to create such belts for a price of around 90 euros without taxes or delivery costs.
- Re-calibrate the rat's side before using it (regarding the spring constants and pulley encoder's reset value).
- Add a piece such that the rat's tail can be secured and away from the treadmill ground.
- Think about the camera's position and the reflective pieces that could impact the recording.

References

[1] Mao YR, Lo WL, Lin Q, Li L, Xiao X, Raghavan P, Huang DF. The Effect of Body Weight Support Treadmill Training on Gait Recovery, Proximal Lower Limb Motor Pattern, and Balance in Patients with Subacute Stroke. Biomed Res Int. 2015;2015:175719. doi: 10.1155/2015/175719. Epub 2015 Nov 16. PMID: 26649295; PMCID: PMC4663281.

[2] Zhao BL, Li WT, Zhou XH, Wu SQ, Cao HS, Bao ZR, An LB. Effective robotic assistive pattern of treadmill training for spinal cord injury in a rat model. Exp Ther Med. 2018 Apr;15(4):3283-3294. doi: 10.3892/etm.2018.5822. Epub 2018 Jan 31. PMID: 29545846; PMCID: PMC5840943.

[3] J. A. Nessler et al., "A robotic device for studying rodent locomotion after spinal cord injury," in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 13, no. 4, pp. 497-506, Dec. 2005, doi: 10.1109/TNSRE.2005.858432.

[4] Propel Physiotherapy, "Body Weight Supported Treadmill Training in Neurological Rehabilitation," [Online]. Available: `https://propelphysiotherapy.com/neurological/body-weight-supported-treadmill-training-in-neurological-rehabilitation/`. [Accessed: Jan. 7, 2025].

[5] Rifton, "Treadmill vs Over-Ground Gait Training," [Online]. Available: `https://www.rifton.com/education-center/articles/treadmill-vs-over-ground-gait-training`. [Accessed: Jan. 7, 2025].

This project utilized ChatGPT for assistance in developing and refining the code, ensuring efficient implementation and optimization of algorithms. ChatGPT served as a *supportive and assistance* tool in the development process as well as a support with grammar checks and structure enhancement for the report.