

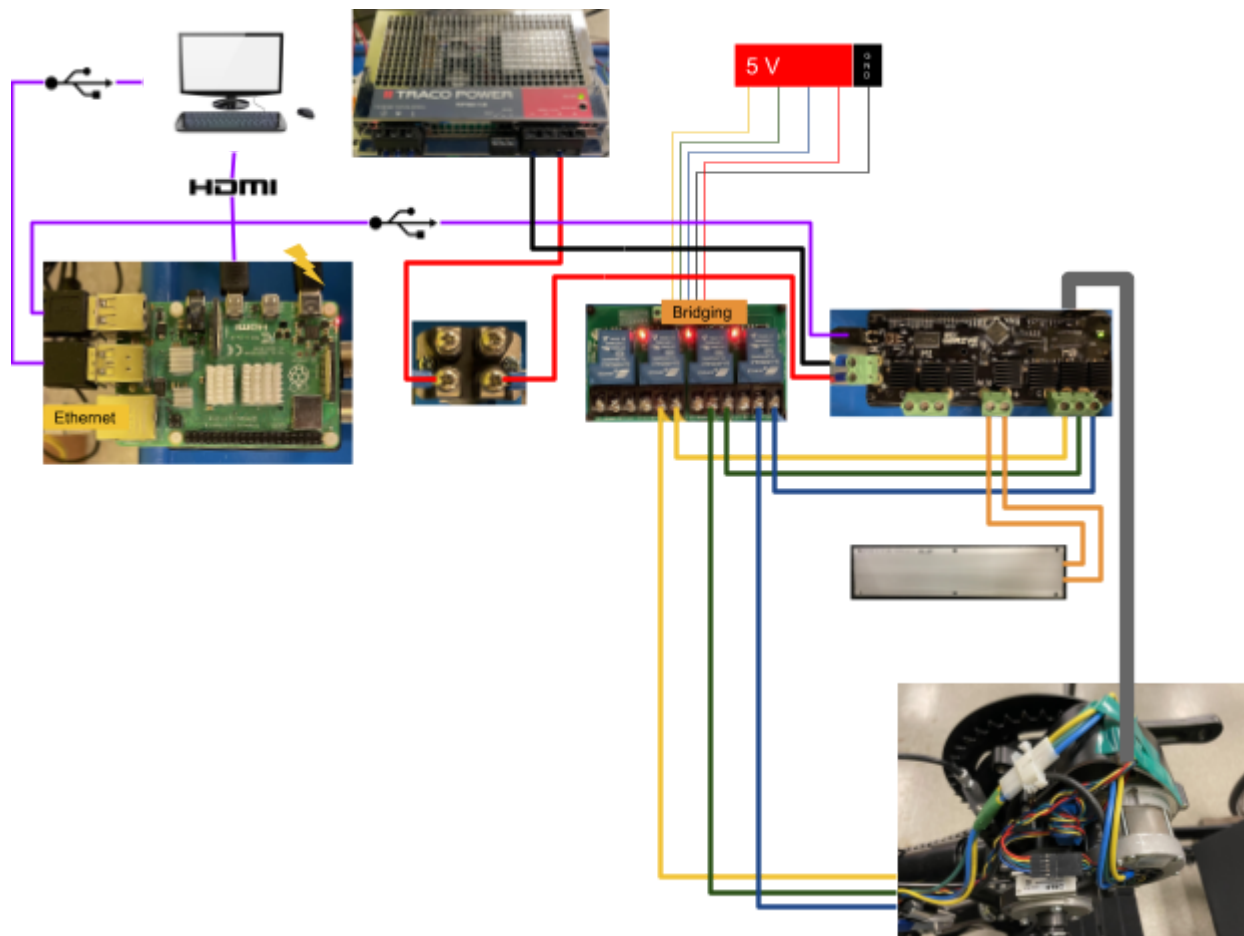
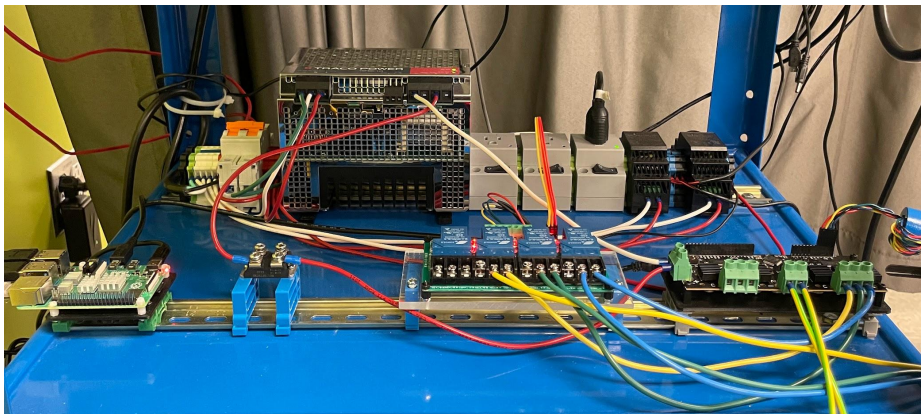
## Ergocycle S2M

## Hardware

If the program is not working, the first thing to do is to check the wiring, some cable may be disconnected with the use.

The paths registered in the “Links and documentation” sections are the path in the git repo <https://github.com/s2mLab/ControlOdrive>.

## Electrical scheme



# Raspberry Pi

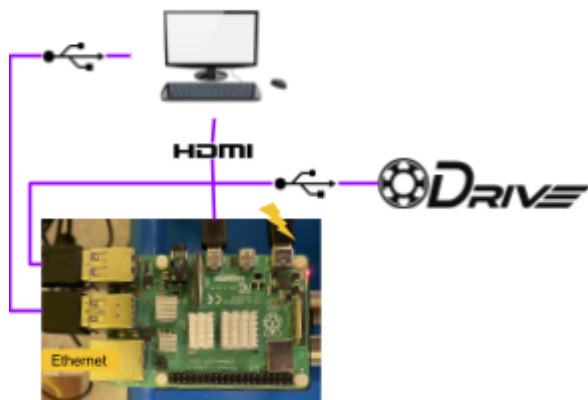
## Model

Raspberry Pi 4 Model B (2018)

## Links and documentation

<https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>

## Connections to the other components



- The board is connected to:
  - power through a mini-usb
  - a keyboard and a mouse by USB
  - a screen by HDMI
  - ethernet
  - the ODrive board by USB

## Divers information

- I used the “Mu” interface during my internship it works well
- I did not find a way for the computer to stay on date but it can easily be reset thanks to ``sudo date MMDDHHmmAA`` if you have a “Your clock is behind error”
- There is a physical watchdog that can be added to the Raspberry and a USB hub if more usb needs to be added.

## ODrive board

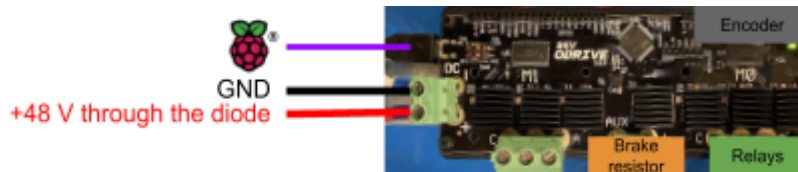
### Model

- ODrive v3.6 56V
- Firmware 0.5.5

## Links and documentation

- Documentation for the code:  
<https://docs.odriverobotics.com/v/0.5.5/getting-started.html>
- Forum: <https://discourse.odriverobotics.com/>

## Connections to the other components



- The board is connected to:
  - The Raspberry Pi board by micro-USB
  - The 48V power supply
  - The brake resistor (the connection order doesn't matter)
  - The relays
  - The encoder:
    - The red wire is for the +5V input
    - The yellow wire is for the A input
    - The green wire is for the B input
    - The blue wire is for the Z input
    - The black wire is for the GND
- If you want to use the `axis1` it is possible but make sure you:
  - Connect the relays to the M1 output
  - Connect the encoder to the M1 input
  - Make the following changes in the code:
    - `axis0` become `axis1`
    - Adapt the numbers of the gpio pins
  - You don't need to change the wiring of the resistor

## Divers information

To try things and see the current errors `odrivetool` is pretty useful.

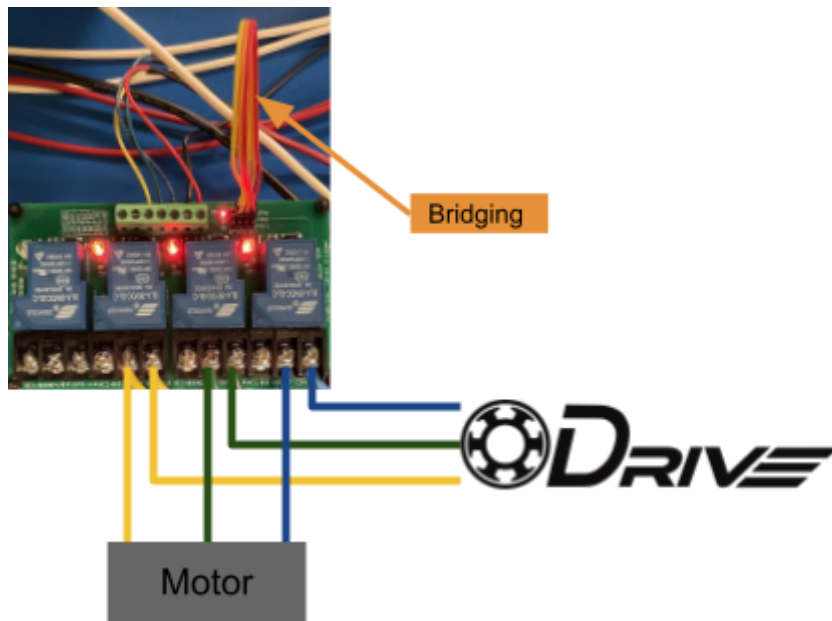
## Relays

It has been chosen to put relays between the ODrive board and the motor to protect the ODrive board from high tensions when it is not supplied. The relays are used as a switch, opened when the board is not supplied, closed else.

## Links and documentation

[ControlOdrive/documentation/relays.pdf](#)

## Connections to the other components



- The relays are alimented by a 5V power supply:
  - The DC+ (reference for the switch) is the red wire connected to +5V
  - The DC- is the black wire connected to the GND
  - The CH1 is the blue wire connected to +5V
  - The CH2 is the green wire connected to +5V
  - The CH3 is the yellow wire connected to +5V
- A bridge is needed between the Com input and the High input, that means that the relays change their state when the channels are supplied by the same tension as DC+. As soon as the power supply is switched on, the CH1, CH2 and CH3 are at the same tension as DC+ so the relays are supplied and then closed.
- The ODrive's wires corresponding to the three phases of the motor are connected to the COM inputs of the relays
- The motor's 3 phases are connected to the NO (Normally Open) inputs of the relays because we don't want any current to flow when the relays are not supplied.

## Diode

It has been advised by one of the crewmembers of ODrive to add a diode between the power supply and the ODrive board in this topic

(<https://discourse.odriverobotics.com/t/power-supply-in-security-mode-when-forcing-on-the-motor/10229>) created by Amandine Chupin .

## Links and documentation

ControlOdrive/documentation/diode.pdf

### Connections to the other components



A diode is a polarized dipole so the connections are obligatory in this direction.

### Brake resistor

#### Model

TE connectivity: TE\_TJT\_500\_3R3\_J\_2040

#### Links and documentation

[ControlOdrive/documentation/brake\\_resistor\\_TE\\_TJT\\_500\\_3R3\\_J\\_2040.pdf](#)

### Connections to the other components



There is no mandatory direction of connection for a resistance.

#### Divers

The resistor kind of leaked when wiring the ODrive to a battery, it seems to still work well now but it wouldn't hurt to check the ohmage sometimes

### Motor

#### Model

Tongsheng TSDZ2, 48V, 500 W

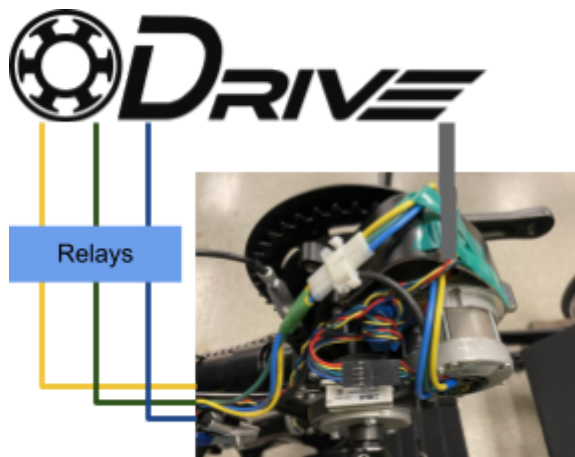
After the motor, there are two gear stages.

- The first one has 8 teeth at the entry and 36 (motor) at the exit (blue gear).
- The second one supposedly has (according to Amandine Chupin's research and checks on the hardware) 10 teeth at the entry and 91 at the exit.

The reduction ratio is then  $Z_e/Z_s = w_s/w_e = 8/36 * 10/91$

The torque constant was not given by the constructor so we calculated it (see Software/Torques)

## Connections to the other components



- The motor is connected to the relays.
- The encoder that is within the motor is connected directly to the ODrive board. The list of the colors is explained in the “ODrive” section.

## Power supply

### Model

Traco Power: TSP\_600\_148

### Links and documentation

[ControlOdrive/documentation/power\\_supply\\_TSP\\_600\\_148.pdf](#)

## Connections to the other components

The power supply is wired to the + of the diode (see “Diode”) and the ground of the ODrive (see “ODrive”).

## Software

If you are not sure what configuration is currently saved in the ODrive it is better to run first the `initial\_calibration.py` file. The pinched noise is totally normal even if it seems unbearable. If no error has been detected, you are good to go!

Run the `main.py` file. You should see the GUI appear.

If you want a simple understanding of the code, you can read and run the `cadence\_control.py` file.

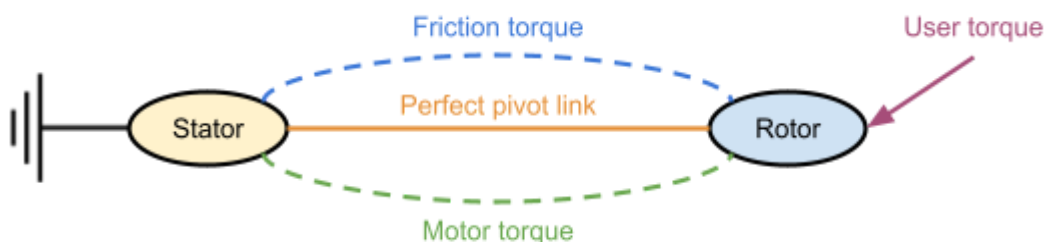
## Versions

2022/02/09:

- Firmware: 0.5.5, *a priori* the last firmware available for the Odrive v3.6 56V (end of life)
  - To update the firmware on the Raspberry, use `sudo odrivetool dfu path/to/firmware/file.elf`
- Software: odrive 0.6.3.post0, the latest version at this time
  - To update the software on the Raspberry, use `pip install odrive==0.6.3.post0`

## Torques

### Equations



The torques are defined as follow:

- $\tau_f$  : Friction torque of the stator on the rotor (we consider the torque due to the hometrainer to be included in this one if the chain is on the pedal's gear)
  - The sign is opposed to the cadence's
  - The intensity depends on the cadence
- $\tau_m$  : Motor torque electrically generated and considered at the output of the pedals
  - The sign depends on the command
  - In theory:  $\tau_m = \frac{1}{r} \eta K_\tau I_m$  with  $K_\tau$  being the torque constant,  $r$  the reduction ratio and  $\eta$  the efficiency factor, since we already consider the resisting torque and we won't be able to calculate  $\eta$  separately from  $K_\tau$  we will consider that  $\tau_m = \frac{1}{r} K_\tau I_m$
- $\tau_u$  : User's torque (torque of the user on the rotor)

- The sign and the intensity depends on the user

Hypotheses:

- These torques are the only torques on the axis of the motor
- $\tau_f$  depends only on the velocity

$$\overrightarrow{\delta_{O,R/S}} = \sum_{i \in \bar{R}} \overrightarrow{\tau_{O(i \rightarrow R)}}$$

The fundamental principles of the dynamics gives:

On the motor axis:

$$\frac{d}{dt}(I_z \omega) = \tau_f + \tau_m + \tau_u$$

Let's simplify this equation by defining the torque of loses:  $\tau_l = \tau_f - \frac{d}{dt}(I_z \omega)$

We have then:  $\tau_l + \tau_m + \tau_u = 0$

First, we consider the system without any user torque:  $\tau_l + \tau_m = 0$

Let's define  $I_l = -I_m$  so that  $\tau_l = \frac{1}{r} K_\tau I_l$

According to some pilot tests,  $I_l$  will be modeled as follow:

$$\begin{cases} I_l \in [-I_0, I_0] & \text{if } \omega = 0 \\ I_l = \text{sign}(\omega)(a\omega + I_0) & \text{otherwise} \end{cases}$$

The parameters  $I_0$  and  $a$  are to be found with a first calibration, without any load and samples at different speeds. I sampled the current during 5 turns at different velocities (from -60 tr/min to 60 tr/min with steps of 5 tr/min). I did a linear regression on the mean of the currents and the velocities at each velocity.

When the cadence is null, we cannot know what is the current corresponding to the resisting torque knowing only the current of the motor. Two examples to support this point:

- The user does not touch the pedals but the motor sends a current inferior to  $I_0$ ,  $I_l = -I_m$

- The user forces in the positive direction with a force inferior to  $I_0$ , and  $I_m = 0$ , then

$$I_l = \frac{-r\tau_u}{K_\tau}$$

In the code we consider that all the current under the  $I_0$  is dissipated by the friction, the rest corresponds to the user torque. In case of the study of a static movement it has to be adapted, because then  $I_l = -\text{sign}(\tau_u)I_0 = \text{sign}(I_m)I_0$ .

Then, we can consider  $I_l$  known and with a known  $\tau_u$  we can identify  $K_\tau$ . It will be the torque constant in the program.

$$\frac{1}{r} K_\tau I_l + \frac{1}{r} K_\tau I_m + \tau_u = 0$$

Hence:



$$K_\tau = \frac{-r\tau_u}{I_l(\omega) + I_m}$$

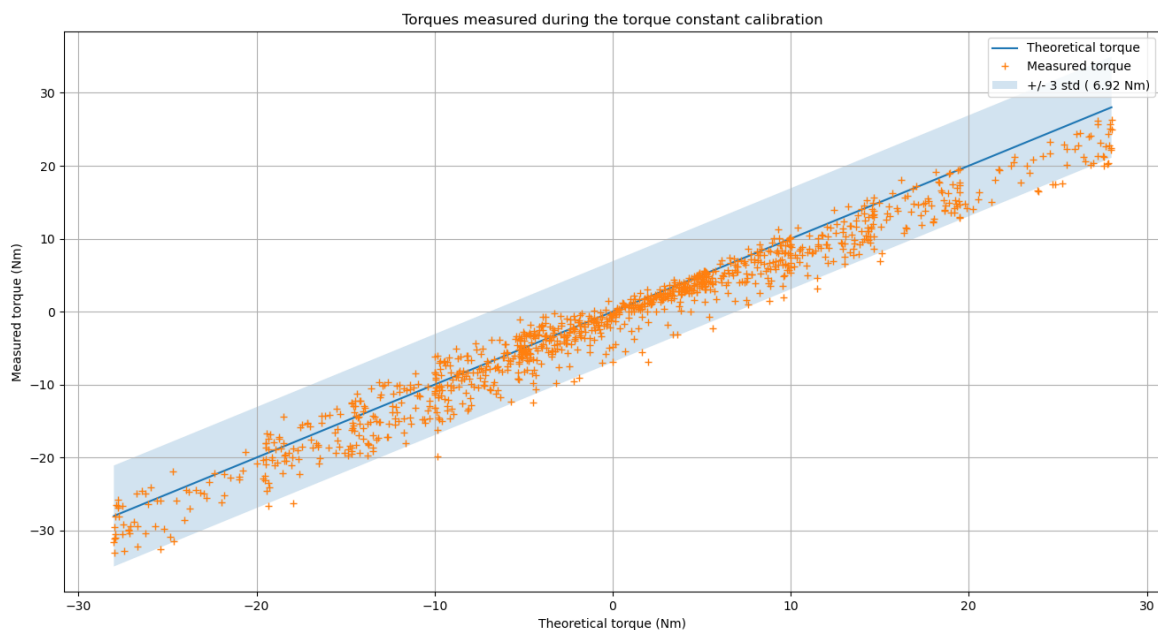
The user torque can be imposed by various methods.

- Impose a torque with an external machine
- Impose a weight at the extremity of the pedals
- Impose a weight at a given distance from the motor axis

The range that interests us is approximately [-45 , 45] N.m (the motor is not capable of more).

### Torque constant estimation

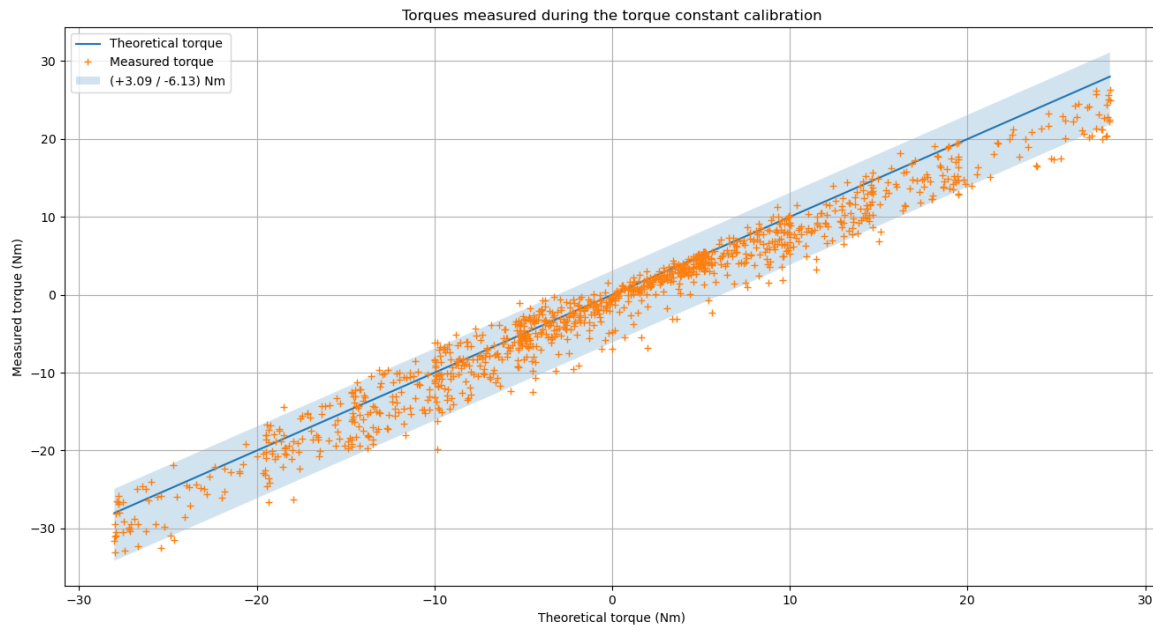
Here are the results of the torque constant calibration.



98% of the measured torques are between the theoretical torque +/- 6.92 Nm.

As can be seen on this graph the measures are mostly below the theoretical curve.

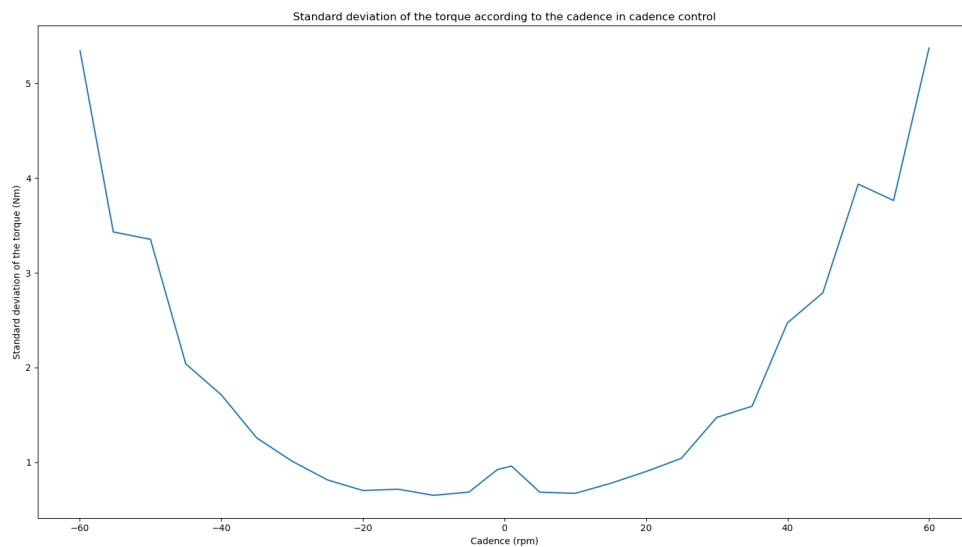
Hence we can define an interval more precise that includes 95 % of the values: the torque + 3.09 / - 6.13 Nm



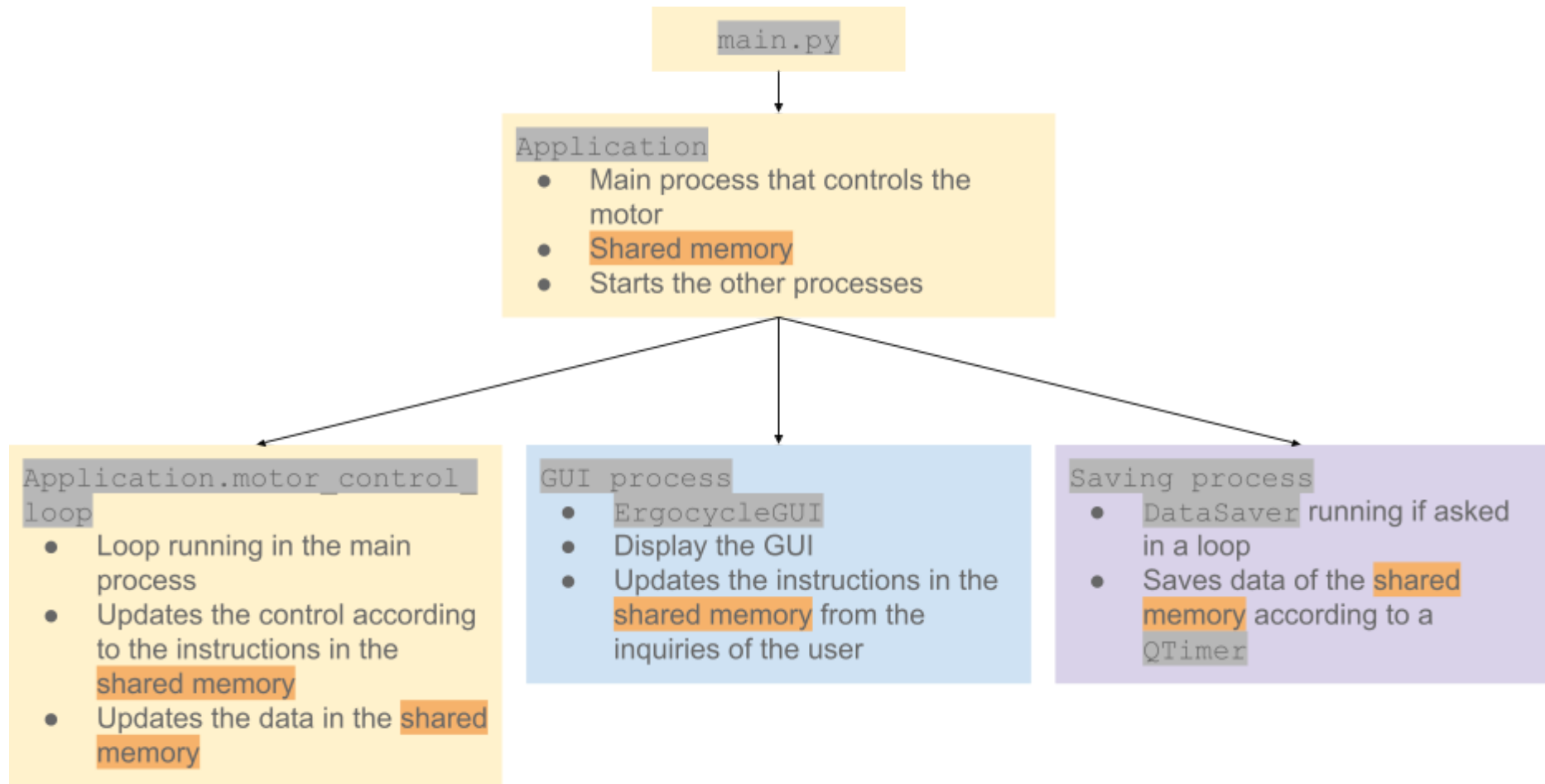
My intuition is that the confidence interval will also depend on the velocity since the shift to the bottom may be a result of the friction

### Torque estimation

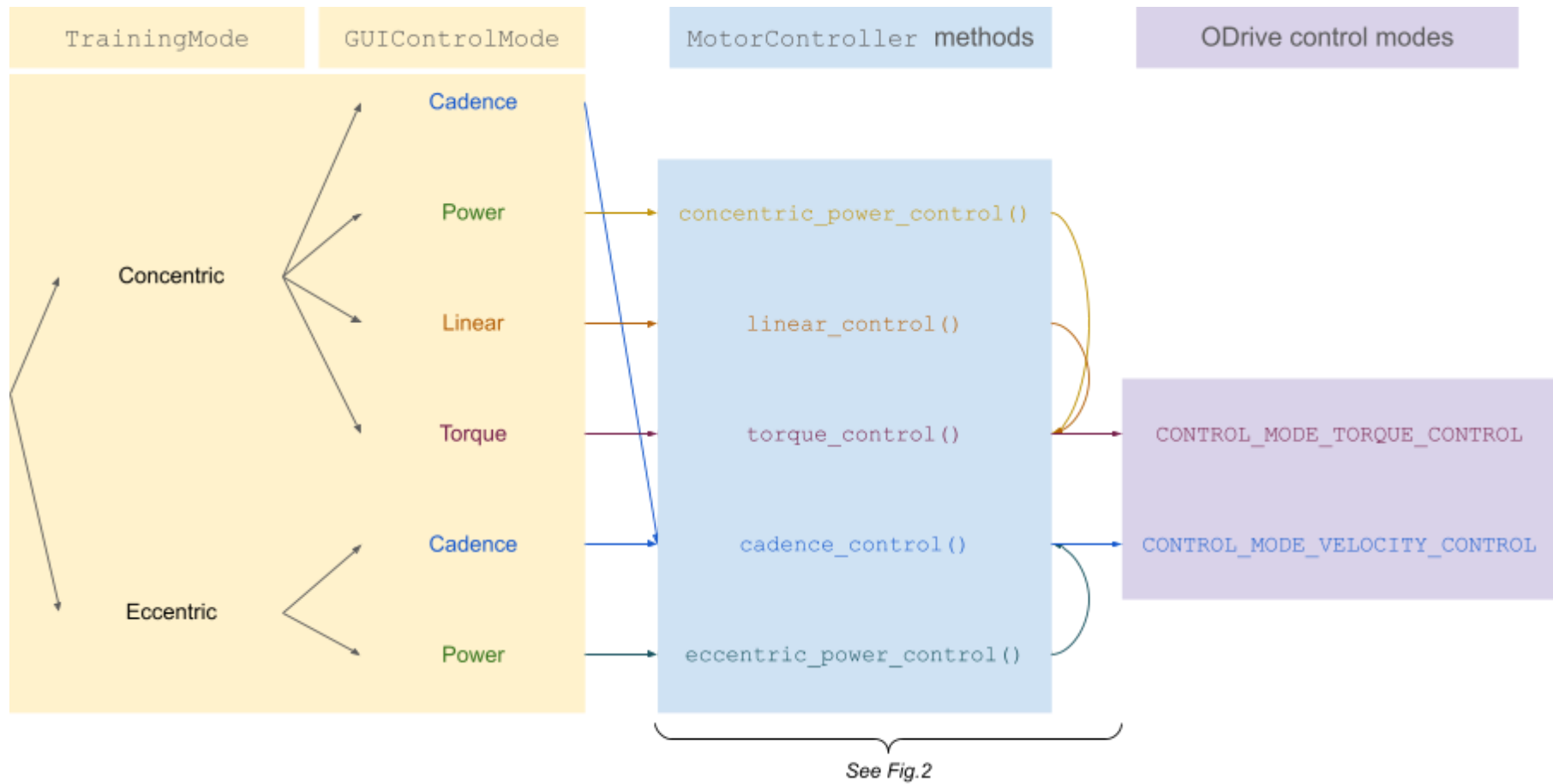
The following graph has been obtained during a resisting torque calibration. It indicates that the higher is the speed, the higher is the noise on the torque.

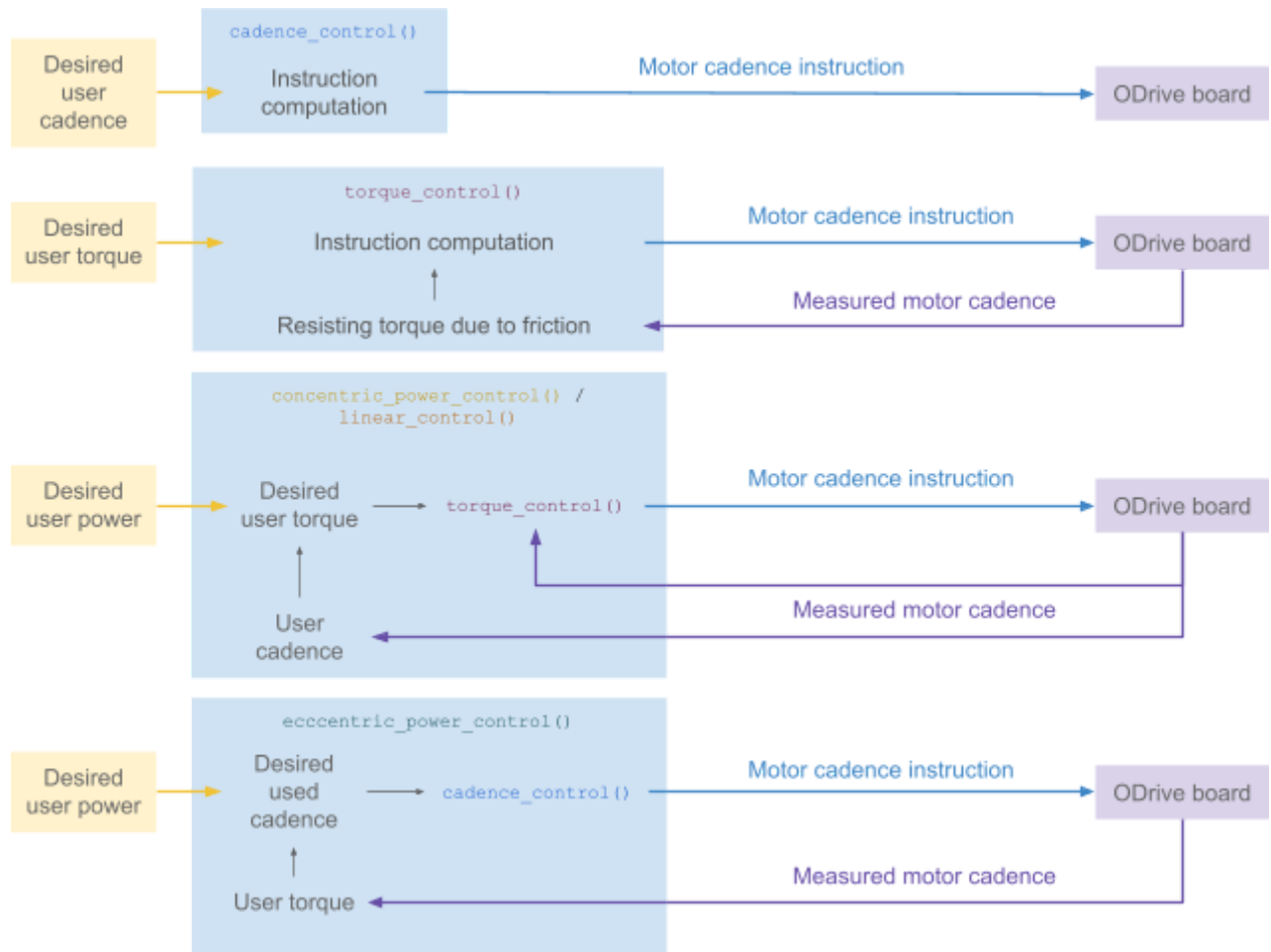


## General structure of ErgocycleS2M



## Control of the motor





## Data saving

The data saving is done in its own process. When the frequency instruction was 10Hz, during the sprints of Kevin Co's experiment done by Amandine Chupin, the period of the saving was (0.09997 +/- 0.0014) s. This is the best result we obtain until now, without the multiprocessing we could only reach +/- 0.02s.

## Limits

There are two types of limits, the limits that are applied to the ODrive (ex: `torque_lim = 100.0`) and the limits in the GUI that are also the limits in the raise (ex: `torque_lim_gui = 40.0`). The latters are meant to protect the user because it cannot completely be protected by the ODrive anymore. The ramp limits are the same in the ODrive and in the GUI.

## Scripts

### ``resisting_current_calibration.py``

As explained in the Software/torque section, the first calibration to do is this one, that will determine the affine relation between the current and the torques of losses.

The calibration will run by itself, make sure that there is no load on the pedals during the whole calibration. It will take several minutes, it is normal. It will modify by itself the file ``hardware_and_security.json``.

### ``torque_constant_calibration.py``

#### Protocol

To determine the torque constant we put weights on one pedal only and make the motor turn at 5 tr/min (because we know the resisting torque when we are in movement and we were not comfortable in going faster). We did it with different weights (3.15, 6.0, 8.80, 11.70, 16.80 kg) and made 3 turns each time.

#### Computation

The computation is done by adapting the script with the new calibration files.

It selects a window of 2 complete turns at a constant velocity (after the ramp) and fits the current \* torque constant to the calculated torque of the weights on the motor (with the weights, the wrench length and the angle).

The calibration has been done twice during Amandine Chupin's internship:

- The first torque constant was 0.11 Nm/A and its coefficient of determination with the data of the second calibration is 0.93. It has been obtained with masses up to 4.5 kg.
- The second torque constant is 0.09 Nm/A and its coefficient of determination with the data of the second calibration is 0.95. It has been obtained with masses up to 16.80 kg.

Both values are registered in ``write_to_json.py``.

### ``resisting_current_gears_calibration.py``

If you want to use the ergocycle with the hometrainer, you must calibrate the torque corresponding to it. For each gear (or one out of two), you need to run this script and pay attention that the file you are writing has the right name (`./calibration_files/resisting_current_gear_{gear_nb}.json``). The easiest gear (largest one) is gear 1 and the hardest gear (smallest one) is gear 10.

### ``resisting_current_gears_computation.py``

Run this script to compute the coefficients that will allow to compute the resisting torque of the hometrainer. We chose to only calibrate the coefficient on the different gear with the

home trainer at 'H' because we didn't find any good relation between the hometrainer levels and the resisting current coefficients.

### **`initial\_calibration.py`**

The initial calibration is a calibration that needs to be done if the motor is changed or if the values of `hardware\_and\_security.json` are changed. It applies these changes and runs the full calibration of the ODrive. It needs to be done without anything touching the pedals and at as little load as possible.

### **`main.py`**

By running this code you will launch the ErgocycleS2M application, e.g. the graphic user interface that allows control of the motor through the ODrive. You can specify the saving period (in seconds) in this file.

### **`cadence\_control.py`**

This script is a tutorial, it is a mere example of how to use MotorController outside of any application and without a graphic interface. It is also saving the experiment data in a file called `cadence\_control\_example.bio` and plotting the cadence.

## Manuel d'utilisation de l'interface graphique

Dans ce manuel, vous trouverez des informations sur chaque élément présent dans l'interface graphique de haut en bas et de gauche à droite.



### Date et heure

La date et l'heure sont celles de l'ordinateur. Si elles ne sont pas correctes, il faut remettre l'ordinateur à l'heure. Les secondes sont affichées ce qui permet de voir si l'interface graphique a cessé de fonctionner.

### Erreurs

En temps normal, il n'y a pas d'erreur, s'il y en a elles seront affichées entre la date et le bouton d'arrêt d'urgence. Pour savoir à quoi elles correspondent, veuillez vous référer à la documentation d'Odrive:

[https://docs.odriverobotics.com/v/0.5.6/fibre\\_types/com\\_odriverobotics\\_ODrive.html#ODrive.Error](https://docs.odriverobotics.com/v/0.5.6/fibre_types/com_odriverobotics_ODrive.html#ODrive.Error)

### Arrêt d'urgence

Le bouton d'arrêt d'urgence arrête l'alimentation du watchdog et donc par extension arrête le moteur. L'interface graphique va également se fermer. Il s'agit d'un arrêt d'urgence en cas de problème. Pour un arrêt classique et plus doux, cliquez sur le bouton "Stop" dans le bloc instructions. Fermer la fenêtre de l'interface graphique a le même effet que le bouton d'arrêt d'urgence.



## Hardware

Gear

0

Il est possible d'installer la chaîne sur le vélo, il faut alors indiquer au programme le pignon actuel pour pouvoir calculer le couple développé par l'utilisateur correctement. Le hometrainer doit être sur la position 'H'. Si la chaîne n'est pas sur le vélo il faut laisser 0. 1 correspond au plus grand pignon (le plus facile) et 10 au plus petit (le plus difficile).

### Bloc d'instruction

Le bloc d'instruction doit être renseigné dans l'ordre suivant: colonne de gauche de haut en bas puis colonne de droite de haut en bas.

Reset the angle and turns	Concentric 2	Power instruction	Torque ramp
0 ° and 0 tr	Power 3	0,00 5 W	5 6 N.m/s
Forward 4	7 Start	8 Stop	

Si vous remplites les instructions dans un ordre qui n'est pas celui-là vous risquez de devoir renseigner plusieurs fois la même chose (par exemple si vous avez renseigné la valeur de l'instruction avant de choisir le mode de contrôle, elle sera remise à zéro en choisissant le mode de contrôle.)

#### 1. Reset angle and turns

Ce bouton est facultatif et toujours disponible. Il permet de définir la position de référence du pédalier par rapport à laquelle vont être calculés les angles et comptes lès tours. Dans le cas où il ne serait pas activé avant le premier 'start' depuis l'ouverture de l'interface graphique, la position de référence sera celle du moteur au moment du 'start'.

#### 2. Mode d'entraînement (concentrique/excentrique)

- Le mode concentrique correspond à une puissance développée positive (sens de rotation et couple appliquée au pédalier dans le même sens). En mode concentrique

sont disponibles les modes de contrôle puissance, cadence, linéaire et couple.



- Le mode excentrique correspond à une puissance développée négative (sens de rotation et couple appliquée au pédalier dans le sens opposé). En mode excentrique ne sont disponibles que les modes de contrôle puissance et cadence.



### 3. Mode de contrôle (puissance/cadence/linéaire/couple)

- Les modes de contrôle en puissance sont différents en entraînement concentrique et en entraînement excentrique.
  - En entraînement concentrique, l'utilisateur va imposer une cadence, le contrôleur Odrive va adapter la résistance du moteur pour être à la puissance cible.
  - En entraînement excentrique, le moteur va se mettre en marche et adapter sa cadence au couple délivré par l'utilisateur.
- En contrôle en cadence, le contrôleur Odrive va chercher à maintenir la cadence du moteur constante quel que soit le couple appliqué par l'utilisateur.
- En contrôle linéaire, l'utilisateur va imposer une cadence, le contrôleur Odrive va imposer un couple de valeur  $\text{instruction} \times \text{cadence}$ . L'instruction étant celle renseignée dans la spinbox 5
- En contrôle en couple, l'utilisateur va imposer une cadence, le contrôleur Odrive va imposer un couple constant déterminé par l'expérimentateur.

### 4. Sens de rotation

Le choix concerne le sens de rotation des pedales avec la nomenclature suivante.



## 5. Instructions

Pour des raisons de sécurité, les instructions sont remises à 0 à chaque changement de mode d'entraînement de contrôle ou changement de sens de rotation. Elles sont toujours positives, leur signe réel étant déterminé par les modes. Leur minimum est toujours 0 mais le maximum ainsi que le pas des flèches dépend du mode.

Mode de contrôle	Maximum	Pas
Puissance	258 W	10 W
Cadence	65 tr/min	10 tr/min
Linéaire	1.0 N.m/(tr/min)	0.1 N.m/(tr/min)
Couple	37 N.m	1 N.m

## 6. Rampes de cadence et de couple

Pour des raisons de sécurité, les instructions sont remises à leur valeur par défaut à chaque changement de mode d'entraînement de contrôle ou changement de sens de rotation. Pour le contrôle en cadence et le contrôle en puissance excentrique, le moteur est piloté en cadence. La rampe correspond au temps que le moteur va mettre atteindre cette cadence cible. Dans le cas d'un contrôle en puissance excentrique, il est conseillé de laisser la pente à 30 (tr/min)/s minimum sinon la cadence est adaptée trop lentement.

Pour le contrôle en couple, linéaire et en puissance concentrique, le moteur est contrôlé en couple. Dans le cas d'un contrôle en puissance concentrique, il est conseillé de laisser cette pente à 5 N.m/s minimum sinon le couple est adapté trop lentement.

## 7. Bouton 'start'

Reset the angle and turns	Concentric	Power instruction	Torque ramp
	Power	0,00 W	5 N.m/s
	0 ° and 0 tr	Forward	Update

Le bouton start permet de lancer le contrôle du moteur.

Dans le cas d'un contrôle en cadence, le moteur va aller à la cadence renseignée. L'utilisateur peut se mettre à pédaler/forcer quand bon lui semble.

Dans le cas d'un contrôle en puissance, le moteur va développer une puissance jusqu'à atteindre les 50 tr/min tant que l'utilisateur ne produit pas une force résistante et dans ce cas la cadence sera adaptée au couple développé par l'utilisateur, avec un maximum de 50 tr/min.

Dans le cas d'un contrôle en couple, linéaire ou en puissance concentrique, l'utilisateur doit se mettre à pédaler après avoir appuyé sur start quand bon lui semble et le contrôle commence à ce moment-là. Dans le cas d'un arrêt du pédalage, la rampe de couple repart de zéro lors du redémarrage.

Un fois le contrôle démarré, le bouton 'start' devient un bouton 'update' et si l'expérimentateur souhaite changer les valeurs de consigne et de rampe, il doit appuyer sur ce bouton pour valider son changement. Le bouton 'reset' de l'angle et du nombre de tours n'est pas soumis au bouton 'update'.

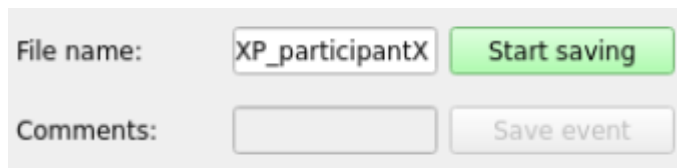
### 8. Bouton 'stop'

Le bouton stop permet d'arrêter le contrôle du moteur.

Dans le cas d'un contrôle en cadence ou en puissance excentrique, le moteur s'arrêtera avec la dernière rampe de décélération donnée en consigne.

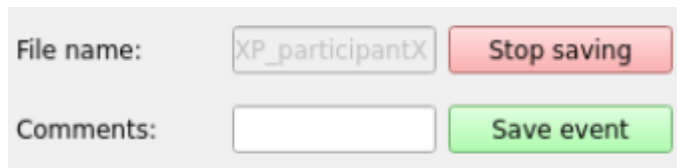
Dans le cas d'un contrôle en couple, linéaire ou en puissance concentrique, une fois que le bouton a été enclenché, l'utilisateur peut s'arrêter de pédaler.

### Bloc d'enregistrement



Le choix du nom du fichier d'enregistrement est laissé libre à l'expérimentateur mais seuls les caractères alphanumériques ainsi que les caractères '-' et '\_' peuvent être utilisés.

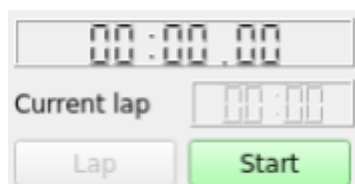
Une fois choisi, l'expérimentateur peut appuyer sur 'Start saving' et l'enregistrement commence. Le fichier sera enregistré en format '.bio' et pourra être lu par le script 'read.py'.



Un fois l'enregistrement démarré, le bouton 'Start saving' devient 'Stop saving' et il est possible de renseigner des commentaires.

Tous les caractères sont disponibles dans les commentaires et ils sont enregistrés au time-code auquel l'utilisateur a appuyé sur 'Save comment'

### Bloc chronomètre

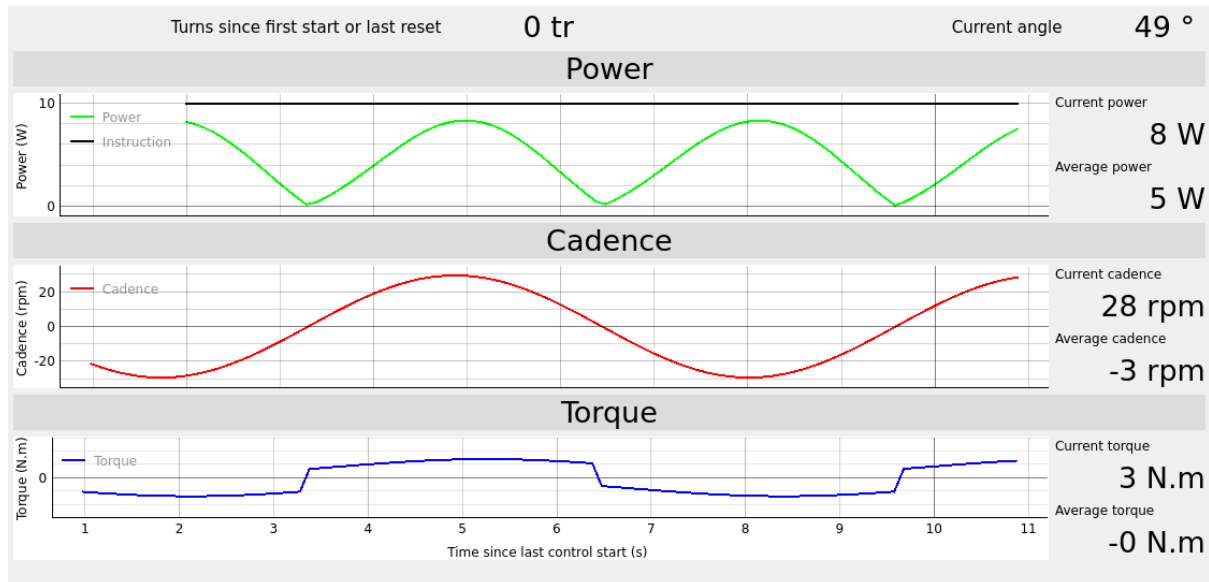


Ce chronomètre n'est pas relié au contrôle du moteur, à l'instar d'un chronomètre physique, il est disponible comme un outil supplémentaire.

Dans le cas d'un enregistrement, sont enregistrés, les temps 'relatif' (depuis le début de l'enregistrement), du chronomètre, et du tour en cours.

Les centièmes de seconde ne sont pas affichés dans le temps du tour. Si le temps dépasse les 99 minutes, le chiffre des centaines n'est pas affiché.

### Bloc visualisation de données



Les données affichées ici sont factices.



Pour le nombre de tour, la cadence et le couple les valeurs sont positives dans le sens indiqué en noir, et négatives dans le sens indiqué en blanc.

Le signe de la puissance est celui du produit de la cadence et du couple, il est positif dans le cas d'un entraînement concentrique, négatif dans le cas d'un entraînement excentrique.

Les angles sont croissants dans le sens noir, décroissants dans le sens blanc.

Dans le cas d'un contrôle en cadence, en couple ou en puissance, une fois le contrôle démarré, une courbe correspondant à la consigne sera également affichée sur le graphique.

## **Idées pour les prochaines versions**

### **Version 2 - Avoir une interface graphique destinée à la personne qui pédale.**

Ajouter une deuxième fenêtre pour la personne qui pedale avec un feed back sur sa cadence, le couple appliqué, la puissance développée et le travail effectué depuis le début. Les features affichées seraient choisies par l'expérimentateur.

### **Version 3 - Lire un fichier qui contient une séquence correspondant à un exercice personnalisé**

Pouvoir charger un exercice prédéterminé et lancer la séquence.

Ex: cadence à 30 tr/min pendant 3 min

Couple a 10 N.m pendant 1 min

...

Avec info sur ou on en est de la séquence + messages adaptés pour l'utilisateur (ex: visez telle cadence).

### **Version 4 - Avoir une interface graphique qui permet de construire un exercice personnalisée**

Faire une interface graphique pour pouvoir construire son propre exercice personnalisé.

### **Features supplémentaires**

- Bouton Help
- Autres formats d'enregistrement
- Ajout de messages pour l'utilisateur
- Clavier tactile
- Raccourcis clavier

## Troubleshooting

### ODrive errors

If your ODrive is not working as expected, run `odrivetool` and type `dump_errors(odrv0)` Enter. This will dump a list of all the errors that are present. To clear all the errors, you can run `odrv0.clear_errors()`.

### Connection to the Odrive

At the beginning, I had trouble connecting the Odrive to the Raspberry Pi. The solution that worked for me was to write in a terminal:

```
echo 'SUBSYSTEM=="usb", ATTR{idVendor}=="1209",
ATTR{idProduct}=="0d[0-9][0-9]", MODE="0666"' | sudo tee
/etc/udev/rules.d/91-odrive.rules
sudo udevadm control --reload-rules
sudo udevadm trigger
```

To test the connection a simple script is sufficient :

```
import odrive
odrive.find_any()
```

Or when launching `odrivetool` in a terminal, it should be written :

```
ODrive control utility v0.5.4
Connected to ODrive 206F3694424D as odrv0
```

**!/ \ If a script using the connection to the Odrive is running, you cannot connect to the Odrive through `odrivetool`. And if `odrivetool` is running in a terminal you can't run a script**

### Motor is not moving at all

The simple test that should work before trying anything else is :

```
odrv0.axis0.requested_state = AXIS_STATE_FULL_CALIBRATION_SEQUENCE
```

### Errors about the current, the resistance, the inductance

Verify you have :

```
resistance_calib_max_voltage > calibration_current * phase_resistance
resistance_calib_max_voltage < 0.5 * vbus_voltage
```

For me (TSDZ2 connected directly to the Odrive), it worked with the following config (which is not the current one in `hardware_and_security.json` it is given if you want to restart from scratch):

```
I_bus_hard_max: inf (float)
I_bus_hard_min: -inf (float)
I_leak_max: 0.10000000149011612 (float)
R_wL_FF_enable: False (bool)
acim_autoflux_attack_gain: 10.0 (float)
acim_autoflux_decay_gain: 1.0 (float)
acim_autoflux_enable: False (bool)
```



```
acim_autoflux_min_Id: 10.0 (float)
acim_gain_min_flux: 10.0 (float)
bEMF_FF_enable: False (bool)
calibration_current: 30.0 (float)
current_control_bandwidth: 100.0 (float)
current_lim: 35.0 (float)
current_lim_margin: 8.0 (float)
dc_calib_tau: 0.20000000298023224 (float)
inverter_temp_limit_lower: 100.0 (float)
inverter_temp_limit_upper: 120.0 (float)
motor_type: 0 (uint8)
phase_inductance: 2.409255648672115e-05 (float)
phase_resistance: 0.04511798173189163 (float)
pole_pairs: 8 (int32)
pre_calibrated: True (bool)
requested_current_range: 25.0 (float)
resistance_calib_max_voltage: 10.0 (float)
torque_constant: 0.20999999344348907 (float)
torque_lim: inf (float)
```

### **Position control and velocity control**

Setting the `odrv0.axis0.controller.config.control_mode`, the `odrv0.axis0.controller.config.input_mode` and the `odrv0.axis0.requested_state` must be done only once and not between each change of position or velocity !

### **Watchdog**

The watchdog timeout can't be as little as the user wants.  
For instance, for a feeding time of 0.01, I've set it to 0.1.

## To do

### Incremental encoder

- Find a way to read the incremental encoder with the ODrive board
  - <https://discourse.odriverobotics.com/t/incremental-magnetic-encoder/10443>
  - <https://discourse.odriverobotics.com/t/incremental-magnetic-encoder-no-response/10505>
  - If it is possible, make sure to change:
    - the mode
    - the reduction ratio (and the pole number, see <https://github.com/ArseneBA/ControlOdrive> for the inspiration)
    - the direction of the encoder, I think it is inverted (not sure)
    - the limits
- For the moment it has been tested to connect the encoder directly to the ODrive board and through the Sensix box and it didn't work (<https://discourse.odriverobotics.com/t/incremental-magnetic-encoder-no-response/10505>) so the following information is just there to give ideas in case it would work one day.

### Model

LM13ICD20DA10A00

LM13 : <https://www.rls.si/eng/lm13-magnetic-linear-and-rotary-encoder-system>

IC : Output type = Incremental, RS422

D20 : Interpolation factor (Resolution) = 200 (10  $\mu$ m)

D : Minimum edge separation = 2  $\mu$ s (0.5 MHz)

A : Reference mark = With reference mark

Magnetic scale or ring must be ordered with reference mark. If required, the cover foil can be installed over

A : Connector = 9 pin D type plug

### Links and documentation

ControlOdrive/documentation/incremental\_encoder\_LM13ICD20DA10A00.pdf

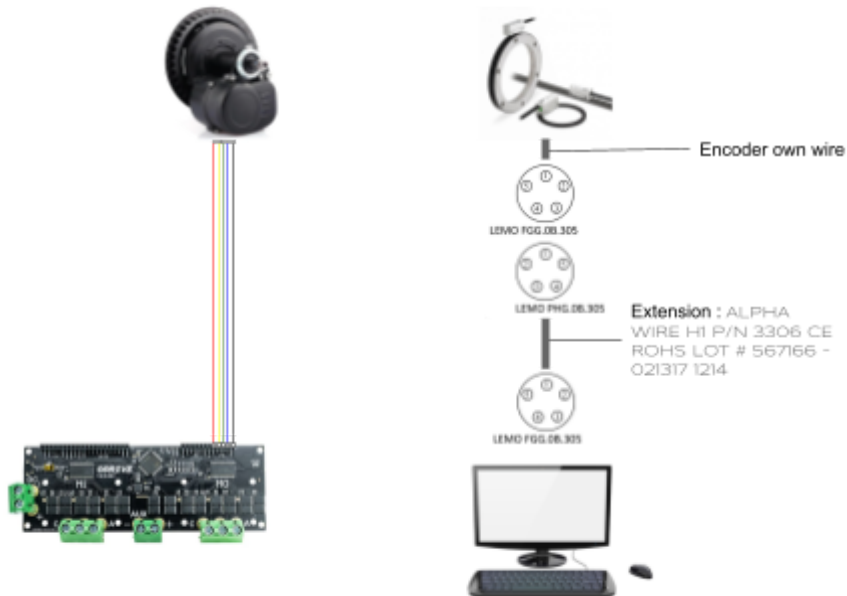
### Why would we want to change the encoder ?

- Improve the precision  
The control was done at 8 kHz on the 2018 ODrive boards. Let's say we are pedaling at 120 rpm -> 2tr/s. There are 48 counts/tr in the hall encoder => 96 Hz << 8 kHz
- Avoid `CONTROLLER_ERROR_SPINOUT_DETECTED`  
(The motor mechanical power and electrical power do not agree. This is usually caused by a slipping encoder or incorrect encoder offset calibration. Check that your

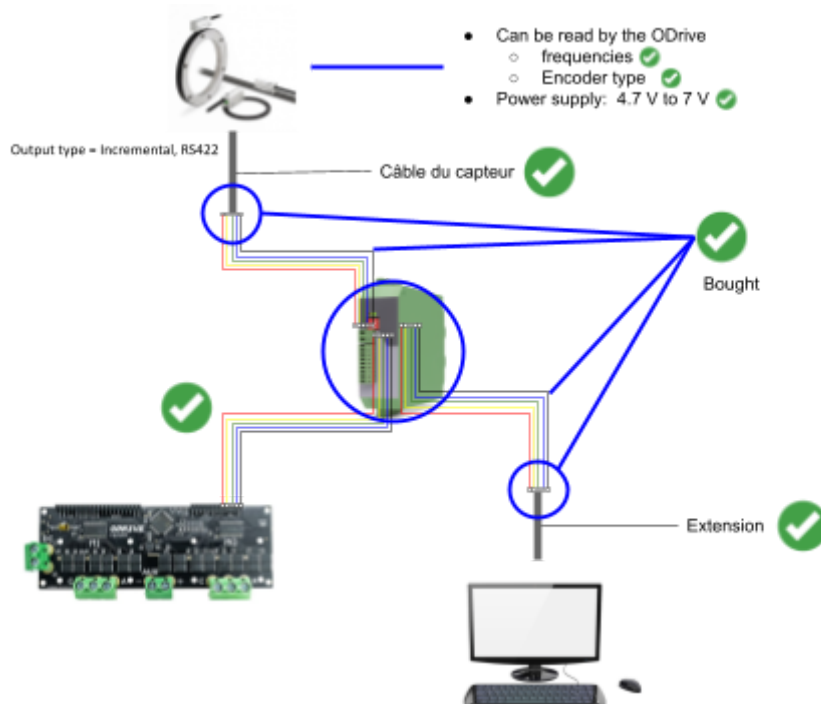
encoder is not slipping on the motor. If using an Index pin, check that you are not getting false index pulses caused by noise. This can happen if you are using unshielded cable for the encoder signals.)

=> It *could* resolve this issue

### Current solution



### Considered solution



## **Cahier des charges**

- Be able to receive angular position information simultaneously with the ODrive and the Sensix box
  - Have the right V+ for the needs of each component
- Maintain the quality of the information delivered by the sensor
  - Wires quality
  - Communication protocols
- Still be able to use the sensor as before (do not modify the existing cables)

## **The best references of splitter I have found**

Main criteria:

- Power supply: 4.7 V to 7 V
- Current consumption <35 mA (without load)
- Incremental, RS422
- 2  $\mu$ s (0.5 MHz)

Splitters:

- Models EAR => frequency too low
- Model GT222 (Pulse splitter with 1 encoder input and 2 potential separated outputs)
  - Doc:  
[https://www.motrona.com/fileadmin/files/bedienungsanleitungen/Gv224\\_f.pdf](https://www.motrona.com/fileadmin/files/bedienungsanleitungen/Gv224_f.pdf)
  - Order (\$565.00, Availability: 4-6 weeks):  
<https://www.enclosureclimatecontrol.com/gt222-motrona-impulse-amplifier-and-splitter-1-input-2-outputs/>
  - Power supply needed (10 - 30 VDC)
  - RS422
  - Max. frequencies: max. 1 MHz at RS422 / TTL
  - Output voltage: 5VDC / 24VDC
  - Output current: max. 500 mA / 250 mA
  - Incremental outputs: At terminal "ext. COM+" the desired output voltage
- Model RXTXD-1-V5-5V-?-5H (RECEIVER-TRANSMITTER UNIT VERSATILE ENCODER INTERFACE)
  - Doc:  
[https://www.encoder.com/hubfs/accessories/Signal-Enhancement/datasheet\\_rtxd.pdf?hsLang=en](https://www.encoder.com/hubfs/accessories/Signal-Enhancement/datasheet_rtxd.pdf?hsLang=en)
  - input 5V okay (no PSU)
  - 5H: TTL, RS422 & RS485 Compatible
  - Frequency Response: Up to 800 Khz
  - Repeater Output Voltage 5V or Vcc
  - Output Current 30 mA/ Channel Max
  - Incremental outputs: At terminal "ext. COM+" the desired output voltage, GND ?

- Modèle SY057
  - Doc: <https://synectic.co.uk/product/encoder-signal-splitter-html/>
  - 612,67 cad
  - Power supply 9-30 VDC
  - 500 KHz
  - The encoder input is switch selectable for operation with either standard RS485/RS422 with differential TTL or HTL signals, Differential voltage or single ended pulse
  - 5 V (no psu but NOT SURE)
  - Incremental outputs: At terminal "+ Lev" the desired output voltage + 0V
  - Current ?

### Torque constant

- Understanding why the relation between current and torque seems affine more then linear
  - If it is because of the friction, find a way to do a dynamic calibration at each velocity
- Prove the precision of our torque constant
- Here are some ideas we had to measure the torque constant
  - Put some weight at the end of a pedal (what we did for our computation)
  - Put some weight at the end of a longer wrench (!\ on the bike the length is limited to 35 cm)
  - Use the ergometer (rigid structure needed)
    - Dynamometric wrench (100 \$ - 200 \$ for the first prices but the minimum torque is relatively high 33 N.m). The best technical solution was at 1200 €.
  - Pedal wrenches equipped with a torque sensor (500 \$)
  - Power sensor (600 \$)

Sensor name	Torque sensor	Dynamometric wrench	
Link	<a href="https://www.a-tech.ca/Product/Series/768/8661_Contactless_Rotary_Torque_Sensor/?tab=2">https://www.a-tech.ca/Product/Series/768/8661_Contactless_Rotary_Torque_Sensor/?tab=2</a>	<a href="https://sensel-measurement.fr/fr/tournevis-et-cles-dynamometriques/2461-diw-cle-dynamometrique-numerique.html#attachments">https://sensel-measurement.fr/fr/tournevis-et-cles-dynamometriques/2461-diw-cle-dynamometrique-numerique.html#attachments</a>	
Scale	Full scale 50 Nm	0,2-75 N.m	
Precision	+/- 0,25% Full scale -> always +/- 0,125 N.m	reading +/-0,5% -> +/- 0,05 Nm at 10 N.m +/- 0,25 N.m at 50	Note: 5 N.m of noise at 60 rpm

		N,m	
USB	USB output and software DigiVision (option)	Option: When outputting data, the USB cable (mini B) is needed	
Installation	It is necessary to fix the motor AND the sensor , to find a piece to couple them, to install all the electrical material. (we measure only the constant of torque and not the mechanical efficiency)	No peculiar installation, the mechanical efficiency will be measured at the same time	

- Calculate the uncertainty on our measures

### Graphic user interface

See 'Manuel d'utilisation de l'interface graphique/Idees pour les prochaines versions'

### Software

- Implement some tests on mock.py

### Small torques and powers

- Find a safe way to help the user. The difficulty lies in the fact that if a torque is applied in the direction of rotation and the user let the pedals go they will turn and it can be really fast

### Cadence control

- Follow the new versions of ODrive: <https://discourse.odriverobotics.com/t/minimal-velocity/10350/2>
- Add a security to limit the torque not to hurt anyone (ex if the cadence is null, not increase the torque to much)

## Symmetric pedaling

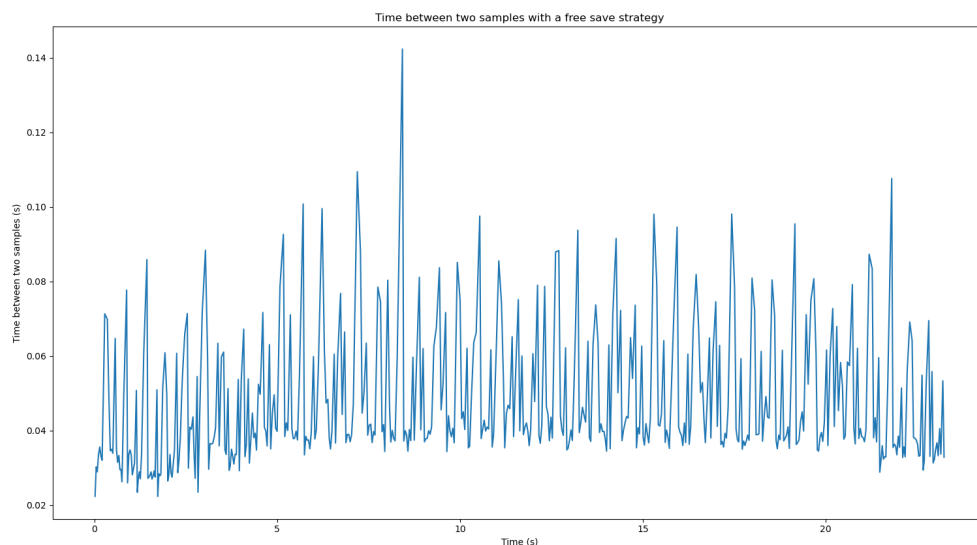
- In the case of a symmetrical pedaling, the resisting torque needs to be recomputed with the angles, it needs a refactor of some functions and a new calibration script.

## Alimentation with the battery

- /!\ The last time we tried it the resistance did not appreciate, here are some of the discussions Amandine Chupin had with the ODrive support if it may help:
  - <https://discourse.odriverobotics.com/t/power-supply-in-security-mode-when-forcing-on-the-motor/10229/7>
  - <https://discourse.odriverobotics.com/t/brake-resistor-overheated-with-battery/10348/2>

## Saving frequency

- If necessary, the saving frequency *could* be better by using `mp.Manager().Queue` instead of `mp.Manager().Value`
- It seems that before the multiprocessing architecture, including the saving in the loop that controlled the motor allowed to achieve a saving period not constant between 0.02 s and 0.14 s. See the branch `saving\_startegies`.



## Sensix pedals

- Draw the hand on the pedals to hurt someone

## Limits and security

- Check all the limits and the security (raise...)

### **Clean closing**

- Currently if the main process is stopped, the GUI window doesn't close. It would be great to check that all the processes terminate at the same time.
  - It works on PyCharm and from a terminal but it doesn't work from Mu
  - See the `cleanup` branch if you want to see what is working from PyCharm and a terminal