Timed running wheel

Clara Hartmann[1], Yves Thönnes and Mahesh M. Karnani[1]

[1]Department of Integrative Neurophysiology
Center for Neurogenomics and Cognitive Research
Vrije Universiteit Amsterdam
De Boelelaan 1085
1081 HV Amsterdam
The Netherlands

General description

The following build instructions are for a timed-access running wheel using cost-effective off-the-shelf components, useful for behavioural neuroscience experiments on rodents (Figure 1). It is controlled by a Raspberry Pi computer that logs degrees of rotation and can stop the wheel using a servomotor break, according to a user-defined Python script. Our example code saves data in a small .csv file containing a list of time stamps (at 5 μs precision using the PiGPIO library[1]) for wheel movement availability, the ensuing first wheel movement and total rotations until the wheel is stopped. A user with beginner skills in Python programming can easily modify the code. The wheel can be easily implemented in a home cage or behavioural setup, as only 150*150 mm floor space and a 10 mm diameter hole are required to pass the cable through.

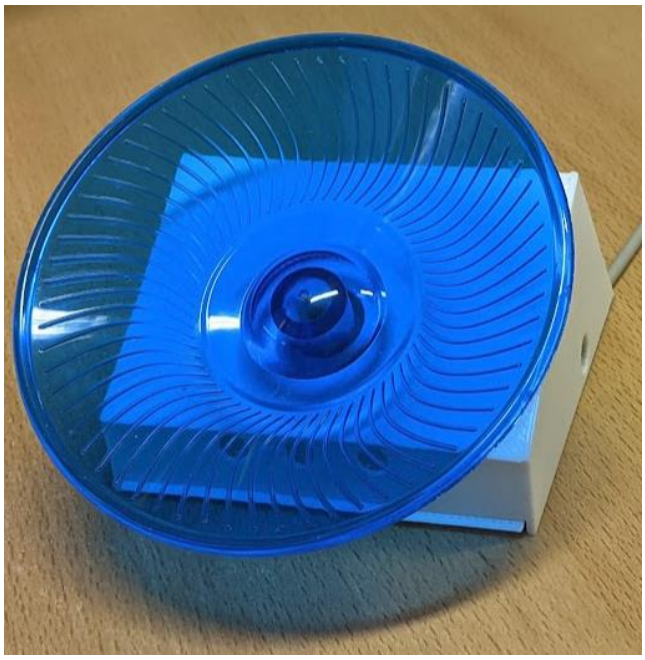

Figure 1, Picture of the timed running wheel.

Materials

Table 1, Bill of materials for the timed running wheel.

Part	Count or length	Supplier/manufacturer	Serial # or *.stl file	Approximate cost, EUR
Arduino nano	1	Arduino	A000005	19.9
Raspberry Pi 400/4B	1	Raspberry Pi	KW-2646	45 – 95
Servo motor (mounting screws and hub attachment included)	1	Master	1556176 - 62	28
Rotary encoder	1	Nidec Copal Electronics	REC16B50-201	20.7
5V power supply	1	RS Pro	124-2183	11.5
4-core shielded cable	2000 mm	Lapp	0034304	1
'flying saucer' running wheel for small rodents	1	Ware pet products	#03281	7
3-d printed base	1		base.stl	1
3-d printed cover	1		cover.stl	1
Rubber tubing 5mm ID/ 7mm OD	20 mm	RS Pro	235-4806	1
Female to female jumper wire connectors	4	MikroElectronika	MIKROE-511	4.3
Male to male jumper wire connectors	4	MikroElectronika	MIKROE-513	4.3
Breadboard	1	Bud industries	BB-32650-B	3.1
			Approx. total cost:	140.4-197.8 EUR

Useful tools: small adjustable wrench, small Phillips head screw driver, wire cutters, soldering iron, superglue.

Mechanical assembly

A plastic 130 mm diameter wheel is removed from a commercially available (e.g., 'flying saucer') rodent running wheel. The shaft of a rotary encoder (Nidec Copal Electronics REC16B50-201) is fitted with a necessary amount of rubber tubing (typically a 20mm long piece of 5mm ID 7mm OD tubing is sufficient) to achieve a tight push-fit on the wheel's axis. 3-D printed parts are found here [2] or in the supporting files and print well on a Prusa MK3 PLA printer. The rotary encoder without the wheel is mounted in the 3-D printed enclosure base (base.stl) along with the 180 deg servomotor (Master servo DS6020 C1689) which acts as the break (see Figure 2 for mounting geometry). A mounting nut and screws are supplied with the components. The supplied short hub of the servo is used, fitted with a soft piece of rubber or foam (e.g., a 20 mm piece of rubber tubing) to dampen contact with the wheel's axis. At this point the wiring should be completed in order to check functionality (see Electronics). After checking function and correct angle of the servo arm (see Adjustment) the servo and rotary encoder and covered with the second level of the enclosure (cover.stl). Lastly the wheel is fitted in place by pushing it onto the encoder's axis, making sure that the wheel can move freely and there is

no friction between the rubber tubing and the enclosure. Superglue can be used to adhere the two parts (base.stl and cover.stl) of the enclosure to each other.

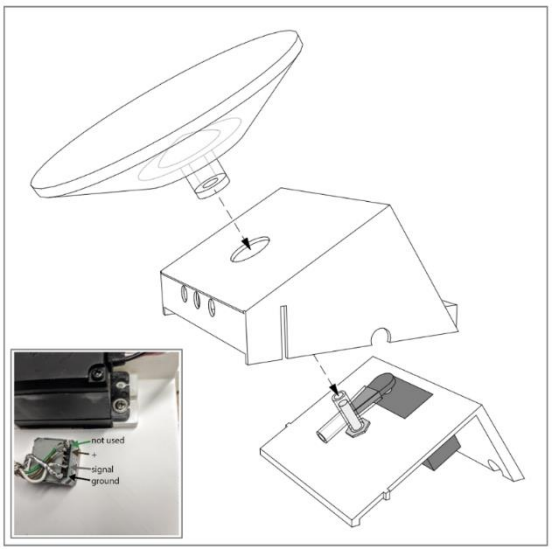

Figure 2, Component mounting geometry.

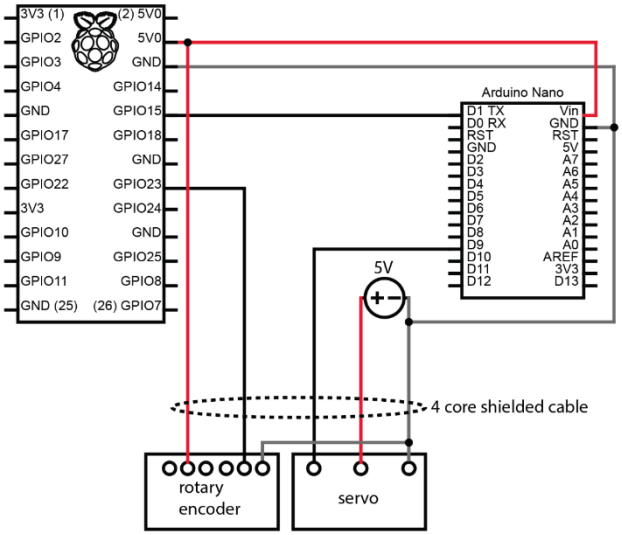

Figure 3, Wiring diagram.

Electronics

A 2 m length of four-core shielded cable is used to connect both the rotary encoder (Nidec Copal Electronics REC16B50-201; 2 leads and ground on pins 2,5 and 6) and 180 deg servomotor (Master servo DS6020 C1689; 2 leads and ground) to a Raspberry Pi, breadboard mounted Arduino Nano and separate 5V DC source (servo). See Figure 3 for connections.

It is important to hide the cable from the rodents. It can be passed through a hole drilled in the floor of the behavioural box below the running wheel, or a square shaped hole can be made in the wall and the built-in cable guide pushed through it.

Code and software set up

An Arduino board is set up to the servo break and a Raspberry Pi to read out and log the rotations. One can set up the software and code following these steps:

1) On the Raspberry Pi (we have used models 400 and 4b) with the operating system installed (Raspberry Pi OS, released 21-02-2023), Arduino IDE (version 1.8.19) is used to install the Servo library.
2) The Arduino code *wheel_only_ard_nano.ino* is downloaded from the supporting files or [3] and flashed to the Arduino Nano. Arduino Uno and Mega boards also work.
3) For setting up Python (Python 3.9.2 is preinstalled on the Raspberry Pi OS), Pandas and Numpy libraries are first installed via the terminal (e.g. `sudo pip3 install numpy` and `sudo pip3 install pandas`). The *requirements.pip* file is downloaded from [4] or supporting files of this document. After navigating in terminal to the folder containing the *requirements.pip* file, all listed requirements are installed by typing `pip3 install -r requirements.pip`.
4) The helper script *Running_wheel_functions.py* and the main script *Generic_wheel_only.py* are downloaded from the supporting files or [4] and placed in the same folder.

5) The PiGPIO daemon is launched from the terminal (`sudo pigpiod`).
6) Now the main script can be run via the terminal or an IDE (we use Thonny Python IDE).

Adjustments

The helper script *Running_wheel_functions.py* can be used to change parameters like how long the wheel should remain open and closed, and where data is stored.
The movement arc of the servo is typically correct with the settings in the sample Arduino code, such that a rubber/foam attachment on the servo arm gently but firmly bends against the axis of the wheel, stopping its rotation. If this is not the case, the angle of the servo in the break and open position must be changed empirically either in the Arduino code lines 8-9 or by manually reinstalling the servo hub at a different angle.

References

1. joan2937/pigpio. https://github.com/joan2937/pigpio.
2. MaheshKarnani. TimedRunningWheel_3Dparts. https://github.com/MaheshKarnani/Switch_maze/tree/main/Modules_SM/TimedRunningWheel/parts_3d_print (2023).
3. MaheshKarnani. Switch_maze_wheel_only_ard. https://github.com/MaheshKarnani/Switch_maze/tree/main/Modules_SM/TimedRunningWheel/wheel_only_ard_nano (2023).
4. MaheshKarnani. Switch_maze_wheel_only. https://github.com/MaheshKarnani/Switch_maze/tree/main/Modules_SM/TimedRunningWheel (2023).