Ecole Polytechnique de Bruxelles

University of Leeds

Johnston Lab Internship

Technical description and User guide: Motion Sensor for behavioral rodent experiments using optical computer mice

Author:
Maxime Verstraeten

Supervisor:
Jamie Johnston

September 16, 2019

Contents

List of Figures

1 Introduction

This user guide intends to provide the user with the information required to use and understand the ball motion sensor device developed at JohnstonLab as well as giving some clues regarding troubleshooting and help with the other tools developed as well.

This device was developed as part of my internship at JohnstonLab (Leeds University, UK) during my master's years as a student engineer from the Université Libre de Bruxelles, Belgium.

This device is part of the rig that allows the lab to perform in-vivo measurements and experiments on mice. This device gathers data from optical computer mice to use them as motion sensor for behavioral studies on mice.

Note: The workflow, thoughts, issues, ... for this project are discussed in my OneNote.

The latest software and documentation updates are available on my Github: https://github.com/maverstr/PS2-Motion-Sensor

2 Objective

When performing behavioral studies on mice, it is interesting to be able to set up a Virtual Reality setup. To do so, the mouse should be able to move freely but still be under the field of view of the microscope, like on a treadmill. The best 3D treadmill is obviously a sphere. This is why for that kind of experiment, the mouse will be running on a Styrofoam ball, held above a bowl by air pressure (this allows minimal resistance between the ball and the bowl). Now to be able to perform virtual reality experimentation, one must know how the mouse, or at least how the ball, moves.

Such device exist on the market but with a very high price.

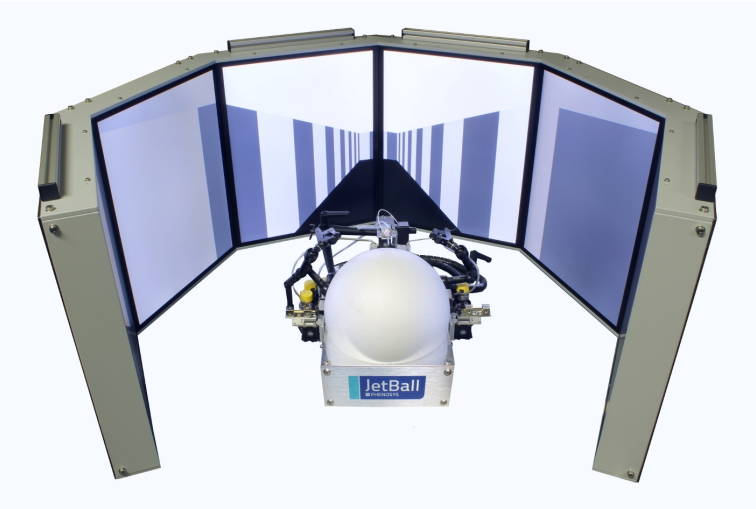

Figure 1: Market VR setup

The goal of this device is therefore to act as a motion sensor for VR experimentation, with a very inexpensive setup.

3 Hardware

The device is extremely simple and is composed of mainly 4 parts:

3.1 Optical mouse

The main element is an optical computer mouse that will act as the sensor. Computer mice have a high sampling rate for computing the deltaX and deltaY and are capable of a great precision.

The main characteristic is the dpi (dot per inch) as it will limit the maximum speed of movement detection and the accuracy. We use a Perixx PERIMICE-201 II (800 dpi and around 7£).

The optical mice can be taken apart to use only the PCB and the lens that come with it, but it requires to do so meticulously. By designing a mouse holder in CAD and keeping the PS/2 connector, it becomes basically "plug&play" and allows easy replacement.

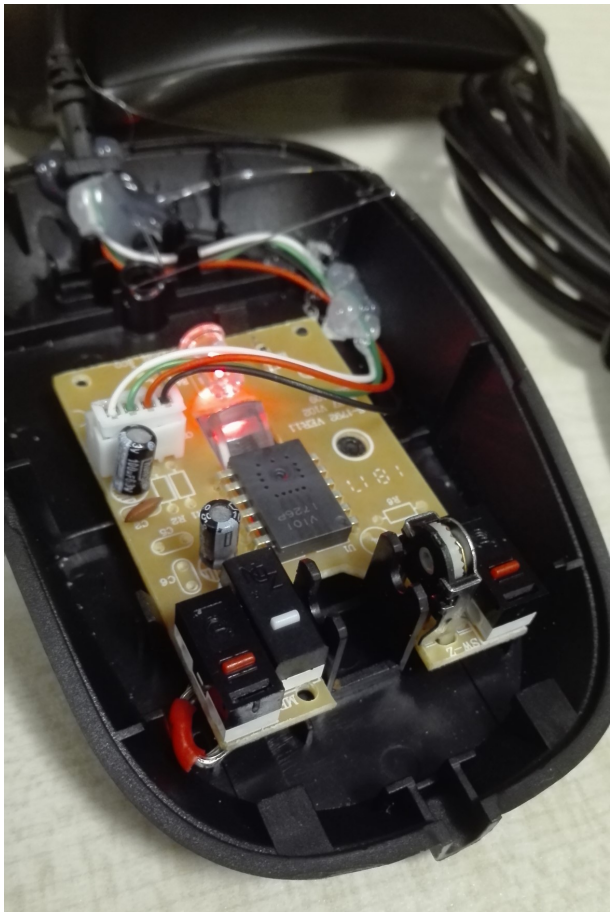

Figure 2: Disassembled mouse

3.2 PS/2 connector

The mice used in everyday life are either USB or PS/2. We want to use the PS/2 connectors as they are easier to "hack" and get data from them.

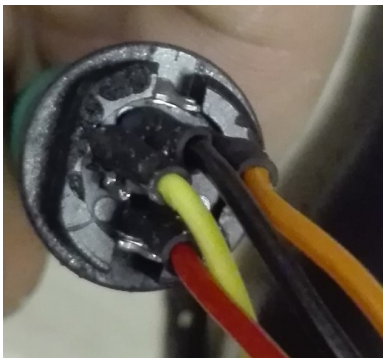

Figure 3: PS2 Connector

These connectors features ground (pin3), 5V(pin 4), SDA (pin 1) and SCL (pin 5). Note that it is reversed for male connectors of course.

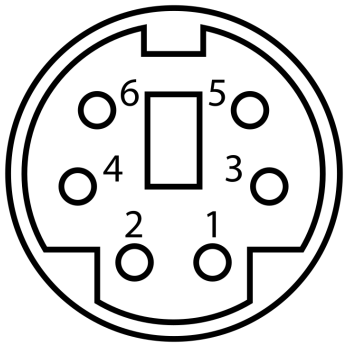

Figure 4: PS2 connector scheme (wikipedia)

3.3 Mouse holder

To hold the mice in place very close to the ball, I designed 3D-printed holders.

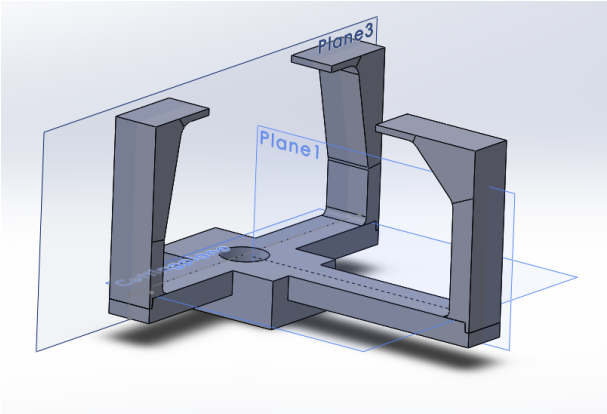

Figure 5: mouse holder CAD design

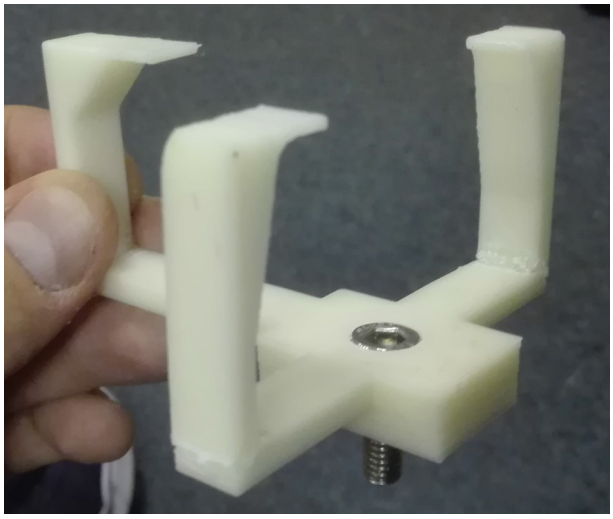

Figure 6: Printed mouse holder

The mouse holders can be attached to the microscopic rig through the bolt going through it.

Figure 7: Mouse held

3.4 Microcontroller

The logic part of the device is handled by an Arduino Mega (even though any Arduino or other μC will do as long as it has 4 digital inputs, 5V supply and a connection to ground). It provides Ground and 5V supply and receives SDA and SCL for each mouse. Note that even though PS/2 is using SDA and SCL, the μC does not require I2C ports with the used libraries. Any digital inputs will do.

4 Software

A PS2 library is able to get the deltaX and deltaY every frame. I simply print the correct values to the Serial and perform vector computation in polar coordinates.

The data can also be transferred over I2C to another μC if needed(but it is not the case in our device plan).

5 How to use?

Simply plug the mice into the hacked PS2 connectors and plug the Arduino. It will immediately start sending the data over Serial.

This data can also easily be transferred over I2C.

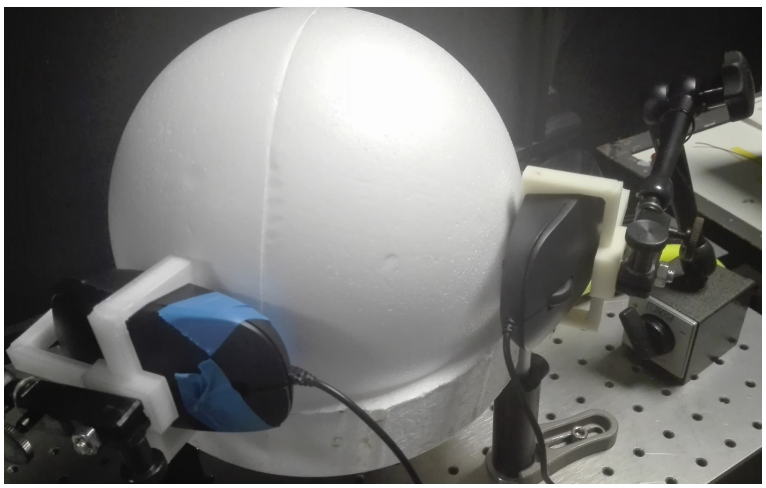

Figure 8: Sensor setup

6 Expected Results

The Arduino should send the motion over Serial. This can be checked through the Arduino IDE. Moreover, the Processing3 script (section 9.1) will draw the corresponding motion on a map (Figure 9).

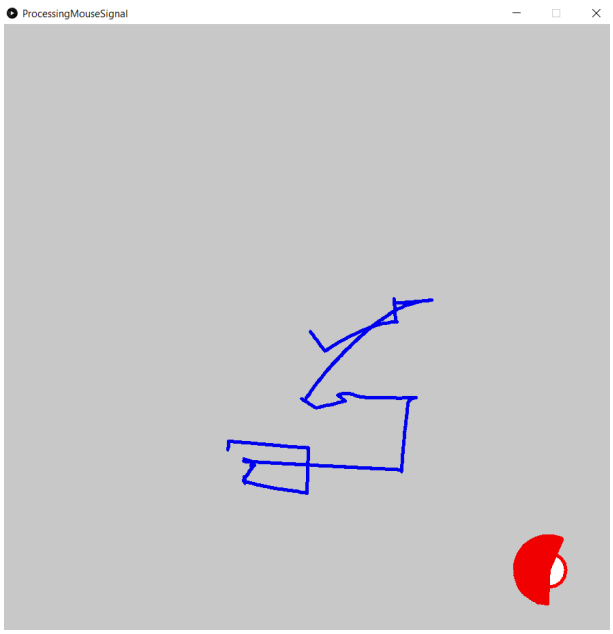

Figure 9: Map drawn on Processing3

7 Troubleshooting

7.1 The motion map does not correspond to actual movements

One mouse handles X and Y movement, the other will handle Z and (redundant) Y. It is therefore important to place them 90 degrees apart from each other as well as perpendicular to the ball.

8 Improvements

8.1 laser mouse

The mouse used for the moment is an optical mouse, which uses a LED as light source which is cheap. The mouse could be replaced by a laser mouse which uses a laser as the source. In both cases, a CMOS is used as the sensor to detect the variation of the surface under the mouse and compute the movement.

Laser mice usually have higher DPI, which means they can track more "dots per inch" and therefore are more sensitive.

Moreover, the optical mice only sense the top of the surface while the laser goes more deeply and see smaller peaks and valleys in the surface. This leads to more jittery data at low speed but a laser mouse is able to work on any surface, even glossy ones (which is somewhat the case with the Styrofoam ball).

8.2 higher DPI

Having a higher DPI is useful to be more sensitive and allow fast movements to be recognized correctly. Indeed, the PS/2 connector will only output 8-bit signed values. So it cannot send more that -127 to +127 every sampling rate (which depends on the mouse and the Arduino code; in early testing the sampling rate was about 60Hz). Having a higher DPI means that the max speed will be reduced as the maximum 8-bit values will be reached on a smaller distance (more dots

per inch) but the mouse will be more sensitive. All in all, a higher DPI means a more sensitive mouse, but one must be careful not to decrease the max speed too much otherwise the data will be saturated all the time.

8.3 Direct use of the sensor

USB mice are already largely prefered over PS/2 mice and it might get difficult in the coming years to find working PS/2 mice. Moreover, the CAD holder is designed for a specific mouse model. It might be interesting to directly wire the connectors to the PCB (easily done thanks to the MTA connectors) and disassemble the mouse or even directly use the sensor. This however requires more work in case of replacement and it might get too technical for a biology lab for such a small improvement. The "plug&play" approach proposed here is therefore preferred.

9 Other tools

9.1 Processing3 Visualisation

I developed a Processing3 (Java) script that gathers data from the Serial port and draws the motion map corresponding to the mouse movement. This allows easy debugging and understanding of how the device works.

To use it, simply plug the Arduino into the computer and select the correct COM port in Processing. A gray window opens, press any key to start and move the mice to observe the drawing.

.1 From mice data to movement mapping

The libraries used allow to get the X and Y data for each mouse as an 8-bit signed value (-127 to +127). Given that the mice are placed 90 degrees apart from each other on the side of the ball (the rodent running on top), a mouse can detects the X movement of the ball (i.e. rotation left/right from a top view of the rodent, perfect crab-walk), the other can detect the Y (i.e. the rodent running forward/backward) and both of them also detect the Z axis movement (i.e. the rodent is turning).

The redundancy of Z can be either discarded on one mouse or used for data verification.

The conversion to an actual map of motion is done by using a rotation matrix using Z as the rotation angle. It is obviously necessary to calibrate the axis (i.e. how many dots for one single 360 degrees rotation. In our case, 13 000 is a good approximation).

We can then decide to either output the absolute values on the Serial and/or to the I2C bus to the microscopic rig and ScanImage or make the computation to output the values with the rotation around the Z axis.

References